VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra Informatiky

Detekce tváří v obrazech

Face Detection in Images

2010

Radovan Fusek

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2010 . . . . . .

Velice rád bych poděkoval doc. Dr. Ing. Eduardu Sojkovi za odborné konzultace, které mi pomohly při vypracování této diplomové práce.

Abstrakt

Cílem práce je prozkoumat dosavadní techniky pro detekování tváři v obrazech. V oblasti detekce tváří existují čtyři kategorie těchto technik. Těmito kategoriemi jsou: přístupy založené na znalostech tváře, rysech tváře, přístupy založené na šablonách a nakonec přístupy založené na zpracování obrazu. Primárním cílem je vytvořit aplikaci, která bude detekci tváří obsluhovat, za pomocí vybrané metody. Jako vybraný přístup je zvolen algoritmus Viola-Jones, který je v textu podrobně popsán. Součástí práce je poskytnutí dosažené úspěšnosti vybraného algoritmu na vhodně zvolených testovacích datech.

Klíčová slova

detekce tváří, Viola-Jones, databáze tváří, AdaBoost

Abstract

The goal of my thesis is to explore existing techniques for face detection in image. There are four technique categories in face detection branch. The categories are: Knowledge-based methods, Feature-invariant approaches, template-matching methods and finally Image-Based methods. The primary purpose is to create an application that will service face detection by usage of the selected method. As the used approach is chosen Viola-Jones algorithm which is described in detail in the thesis. The part of the thesis is to provide fruitfulness of the chosen algorithm with usage of properly selected test data.

Keywords

Face Detection, Viola-Jones, Face Database, AdaBoost

Seznam použitých symbolů a zkratek

RGB	RGB je barevný prostor složený ze tří barevných kanálů Red (červená), Green (zelená), Blue (modrá)
HSV	Barevný model určující barvy jejich odstínem, sytostí a jasem
CPU	Procesor, výkonná jednotka počítače
RAM	Paměť s libovolným přístupem
YCrCb	Barevný model, Y představuje jasovou složku, CB modrou složku a CR červenou složku
YIQ	Model používaný pro přenos televizních signálů v normě NTSC
HMM	Hidden Markov Model
API	Sbírka procedur, funkcí, tříd, které může uživatel knihovny, programu používat
GUI	Grafické uživatelské rozhraní
SVM	support vector machines

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Rozpoznávání obrazů nebo vzorů se jako studijní obor významně vyvinul v šedesátých letech 20. století. Obor zasahoval do mnoha odvětví a přebíral vývojové trendy v oblasti statistiky, umělé inteligence, informatiky, matematiky a mnoha dalších. Rozpoznání vzorů pochází z potřeby pro automatizované rozpoznávání objektů, signálů a obrazů nebo z potřeby automatizovaného rozhodování založeného na dané sadě parametrů.

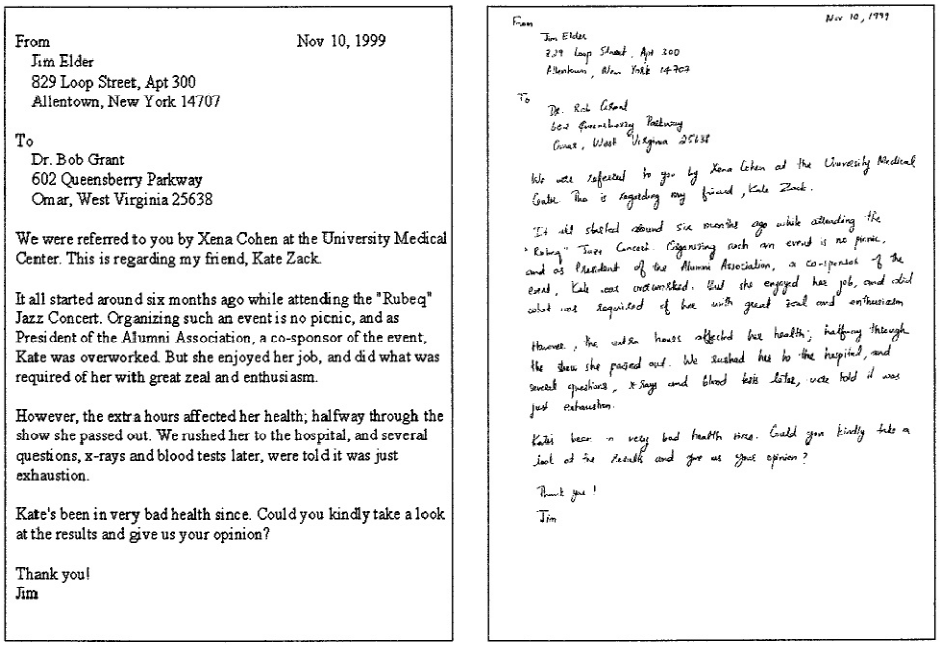
Navzdory tomu, že vývoj v tomto oboru trvá přes půl století, jedná se stále o aktivní oblast výzkumu i kvůli rychle rostoucímu počtu aplikací, kterým může rozpoznání vzorů přinášet prospěch. Základní výzva v automatizovaném rozpoznávání a rozhodování je skutečnost, že problém rozpoznání obrazců, který se zdá být jednoduchý dokonce pro dítě, je ve skutečnosti daleko obtížnější při převodu do strojové domény.

Pro jednoduchou demonstraci (kterou uvádí [18]) se dá považovat například problém rozpoznávání pohlaví (obrázek 1). Pro lidi je tento úkol poměrně snadný a bez námahy dokážou pohlaví osob rozeznat.

Obrázek 1: Problém rozpoznávání pohlaví [19]

Pokud by, ale byla snaha takový problém převést do strojové domény a rozhodnutí o pohlaví nechat na stroji, dostaví se celá řada méně či více složitých otázek. Jaké charakteristické rysy jsou mezi těmito dvěma třídami (muži, ženy), aby se stroj mohl inteligentně rozhodnout? Určitě přijdou na mysl vlastnosti jako vlasová délka, poměr mezi výškou a váhou, tělesné

zakřivení, obličejové výrazy nebo obličejová struktura kostí. Ani kombinace těchto vlastností nemusí nutně vést stroj ke správnému určení pohlaví. Ačkoliv člověk může bez větší námahy identifikovat pohlaví osob, ve strojové doméně tomu tak zdaleka není a je potřeba věnovat specifikaci vlastností, díky kterým se má stroj rozhodnou velkou část práce. Postupem času takové problémy našly uplatnění v řadě aplikací a vědních oborů.



From Nov 10, 1999
Jim Elder
829 Loop Street, Apt 300
Allentown, New York 14707

To
Dr. Bob Grant
602 Queensberry Parkway
Omar, West Virginia 25638

We were referred to you by Xena Cohen at the University Medical Center. This is regarding my friend, Kate Zack.

It all started around six months ago while attending the "Rubeq" Jazz Concert. Organizing such an event is no picnic, and as President of the Alumni Association, a co-sponsor of the event, Kate was overworked. But she enjoyed her job, and did what was required of her with great zeal and enthusiasm.

However, the extra hours affected her health; halfway through the show she passed out. We rushed her to the hospital, and several questions, x-rays and blood tests later, were told it was just exhaustion.

Kate's been in very bad health since. Could you kindly take a look at the results and give us your opinion?

Thank you!
Jim

Obrázek 2: Rozpoznání ručně psaného textu [20]

Může se jednat například o rozbor písma (obrázek 2), kdy ručně psané dokumenty požadujeme převést do digitální podoby nebo o rozpoznávání otisků prstů (obrázek 3), které najde obrovské uplatnění v kriminalistice.

Obrázek 3: Rozpoznávání otisků prstů [21]

Rozpoznávání obrazů se také v dnešní době hojně využívá v medicíně pro rozbor mikroskopických dat, vyhodnocování rentgenových snímků, atd. Své místo si rozpoznání našlo i v průmyslu, vojenství a dokonce i ve zpracování satelitních snímků (obrázek 4).

Obrázek 4: Rozpoznávání budov ze satelitních snímků [20]

Nutno podotknout, že rozpoznávání jako takové se nemusí nutně týkat jenom obrazů, ale například i zvuků. Text je, ale zaměřen pouze na oblast využití v obrazech.

Ve druhé kapitole jsou představeny hlavní výzvy a problémy při detekci tváří v obrazech a další možné využití samotné detekce. Také zde budou představeny čtyři existující kategorie detekčních technik pro odhalení tváře. Bude se jednat o techniky založené na znalostech tváře, rysech, šablonách a na zpracování obrazu. Po tomto představení jednotlivých technik, bude v sedmé kapitole představen vybraný přístup pro praktickou realizaci. Tímto přístupem je Viola-Jones detektor založený na Haarových příznacích, integrálním obraze a na výcvikovém algoritmu AdaBoost. Všechny jeho části jsou také v textu popsány. Jedna z kapitol je věnována i samotné implementaci aplikace. V posledních kapitolách bude představen postup trénování klasifikátoru, který bude použit pro detekci. Klasifikátorů bude představeno hned několik. Nakonec bude provedena série testů jednotlivých klasifikátorů.

2. Rozpoznávání tváří

Výzkum detekce tváří patří do oblasti zpracování obrazu, která se zabývá rozpoznáváním vzorů a počítačového vidění. Detekce tváří a jejich rozpoznání jsou prvními kroky k široké škále možných aplikací založených na rozpoznávání identity. Nalezení tváře v obraze podléhá mnoha různým faktorům ovlivňujícím úspěšnou detekci. V daném obraze je potřeba nalézt tváře a například je označit. Naopak vše ostatní, co tváří být nemá, je potřeba ponechat bez označení. Bohužel ztěžuje situaci fakt, že tváře jsou objekty, které se vyznačují značnou proměnlivostí. Například existují lidé s různou barvou pleti, různým tvarem tváře nebo s různou velikostí tváře. Proto je snaha, aby automatizovaný systém, který by detekci obsluhoval, byl vůči těmto změnám imunní. Jinými slovy je potřeba, aby výsledný detektor označil i tváře, které mohou být určitým způsobem těmito faktory ovlivněny. Takovými faktory mohou být:

- **Póza (postoj, držení těla)**

Tváře v obraze se mění podle různých postojů, které daní lidé na obrazech mají. Lidská tvář může byt zachycena čelně, z profilu, ale i ze spodu. Díky tomu mohou být určité obličejové rysy jako nos nebo oči zcela zakryty.

- **Přítomnost či nepřítomnost obličejových doplňků, rysů**

Do kategorie obličejových doplňků mohou spadat například fousy, kníry nebo dokonce i brýle. Jejich barva, velikost, umístění jsou další faktory ovlivňující detekci. Problém obličejových doplňků ilustruje obrázek 5.

Obrázek 5: Vliv změny obličejových doplňků [16]

- **Výraz obličeje**

Výraz tváře se považuje za další ovlivňující faktor, který může změnit geometrii obličeje. Lidé mohou mít na snímcích různou náladu a podle toho se samozřejmě mění i jejich výraz. Problém změny výrazu tváře ilustruje obrázek 6.

Obrázek 6: Vliv změny výrazu [16]

- **Zakrývání**

Tváře mohou být částečně zakryty jinými objekty nebo dokonce i jinými tvářemi. Tato situace může také značně ovlivnit samotnou detekci. V okamžiku, kdy je na obraze zobrazena například jen část tváře, může nastat problém s jejím nalezením.

- **Vnější podměty**

Při vytváření obrazu hrají v detekci roli faktory, jakými jsou osvětlení, kamerové vlastnosti a jiné. Problém změny osvětlení ilustruje obrázek 7. Na obrázku je vidět, jak stejná tvář mající statický výraz, zobrazená ze stejného úhlu může vypadat pod různým typem osvětlení odlišně.

Obrázek 7: Vliv změny osvětlení [16]

- **Velikost tváře**

V obraze se tváře mohou vyskytovat v různých velikostech. Samozřejmě, že po systému, který by detekci vykonával, budeme chtít, aby detekoval tváře i v různých velikostech. Problém může nastat v případě, kdy je tvář na snímku příliš malá. V takové situaci může být složitější extrahovat důležité rysy tváře a tvář v obraze nemusí být nalezena (problematiku velikosti tváře demonstruje obrázek 8, kde se v pozadí vyskytují tváře menších velikostí než v popředí).

Obrázek 8: Ukázka různých velikostí tváří v obraze. Obrázek je obsažen v testovací sadě CMU[24]

2.1 Hlavní zkoumané okruhy v oblasti analýzy tváří

S rozvojem automatických systémů se zvednul i počet užitečných aplikací týkajících se oblasti analýzy tváří. Mezi tyto aplikace patři:

- **Rozpoznání tváře** (Face recognition): Spočívá v rozpoznání lidí v obrazech. Jinými slovy, chceme přiřadit jednu jménu jednu tvář. Taková aplikace poté může být použita například v bezpečnostních systémech.

- **Lokalizace tváře** (Face localization): je problém týkající určení přesné pozice tváře v jednotlivých obrazech. V okamžiku, kdy jsou již zjištěny tváře v obrazech, může být snaha je přesně lokalizovat.

- **Sledování tváře** (Face tracking): má za cíl sledovat detekovanou tvář v sekvenci obrazů. Toho může být často využito v kamerových systémech. Pokud kamera zachytí určitou tvář, zaměří se na ni a začne ji sledovat.

- **Výraz tváře** (Facial expression): motivací může být rozpoznání citových stavů detekovaných obličejů (štěstí, smutek).

Prvním krokem pro všechny tyto problémy je nalezení tváří v obrazech. První úsilí detekovat tváře se datuje od 70. let 20. století, kdy vznikaly jednoduché heuristické a antropometrické techniky. Tyto techniky velmi lpěly na dodržování prostého pozadí obrazů nebo na čelním pohledu tváře. V důsledku toho, byly omezeny jen na práci s typickými pasovými fotografiemi. Od 90. let 20. století je datován další růst výzkumu v oblasti lokalizace a rozpoznání tváří. Za použití statistiky a neuronových sítí bylo možné popsat další efektivní metody detekování tváří. Do dnešního dne bylo popsáno kolem 150 přístupů jak nalezení tváře dosáhnout.

Další sekce podrobně vylíčí přední přístupy pro odhalení tváře. Přístupy založené na znalostech tváře, přístupy založené na rysech tváře, přístupy založené na šablonách a přístupy založené na zpracování obrazu.

2.2 Metody pro detekci tváří

Existuje několik přístupů pro detekci tváří, které by se daly rozdělit do čtyř hlavních kategorií. Těmito kategoriemi jsou:

- **Přístupy založené na znalostech tváře (Knowledge - Based):** Tyto přístupy se pokoušejí tvář zakódovat do určitých popisných pravidel, do kterých zakódují informace o tvářích. Obvykle pravidla zachycují vztahy mezi jednotlivými rysy tváře. Tyto metody jsou navrženy především pro lokalizaci tváří.

- **Přístupy založené na rysech tváře (Feature - Based):** Tyto přístupy se pokoušejí nalézt rysy tváří, které by nebyly ovlivněny například osvětlením nebo výrazy ve tvářích. Podle těchto nalezených rysů mohou být tváře lokalizovány.

- **Přístupy založené na šablonách (Template - Based):** Tyto přístupy počítají vztahy mezi standardním vzorem tváře (šablonou) a vstupním obrazem. Standardní vzory tváří bývají předem známy.

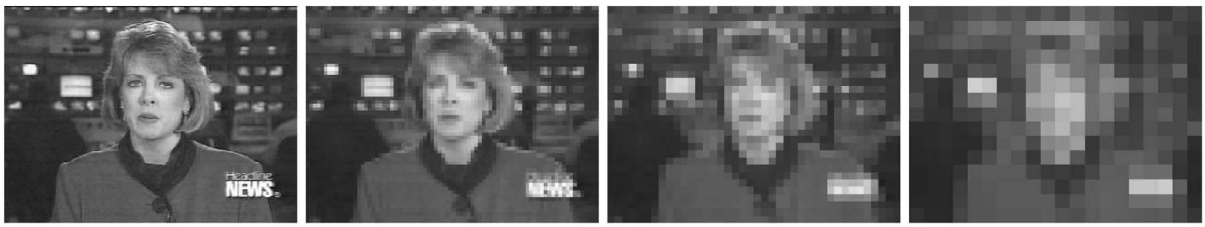
- **Přístupy založené na zpracování obrazu (Image - Based):** Tyto přístupy využívají na rozdíl od přístupů založených na šablonách výcvikovou sadu tváří pro vytvoření modelu tváře s cílem, že různorodost tváří ve výcvikové sadě by mohla zachytit i proměnlivosti lidských obličejů.

V dalších sekcích budou jednotlivé kategorie více přiblíženy. V každé kategorii bude popsán minimálně jeden její zástupce.

3. Přístupy založené na znalostech tváře (Knowledge - Based)

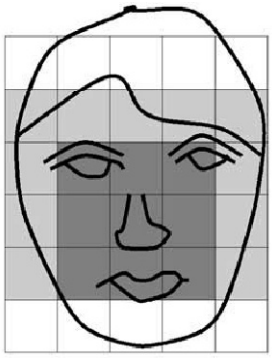
Metody na bázi takového přístupu jsou založeny na pravidlech odvozených díky zkoumání lidských tváří. Jedná se o jednoduchá pravidla, která popisují rysy lidské tváře a jejich vztahy. Může se jednat například o souměrnost očí, nosu nebo úst. Vztahy mezi jednotlivými rysy mohou být popsány jejich vzdálenostmi a pozicemi. Proto jsou nejprve extrahovány z obrazu obličejové rysy a identifikace tváře probíhá na základě zakódovaných pravidel.

Jeden přístup představili ve své práci Yang a Huang. Jejich systém se skládá z tří úrovní pravidel. Na nejvyšší úrovni jsou nalezeny všichni tzv. kandidáti na tvář. Pravidla v této úrovni popisují především tvar tváře. Naproti tomu v dalších úrovních se pravidla spoléhají na podrobnost obličejových rysů. Na první úrovní jsou tedy vytvořeny hierarchie obrazů z originálního obrazu. Obrazy v hierarchii postupně ztrácí svoji rozlišovací schopnost (obrázek 9).



Obrázek 9: Ukázka postupného snižování rozlišovací informace v obraze. Na levé straně je originální obrázek. [1]

Na takto upravené obrazy může být aplikováno následující pravidlo. Centrální část tváře (tmavě stínované části na obrázku 10) a okrajové kulaté části tváře (světleji stínované části na obrázku 10) mají v nízko rozlišovacích snímcích jednotnou intenzitu jasu. Ale rozdíl jasů mezi těmito částmi může být významný.



Obrázek 10: Rozdělení obličeje do několika regionů [1]

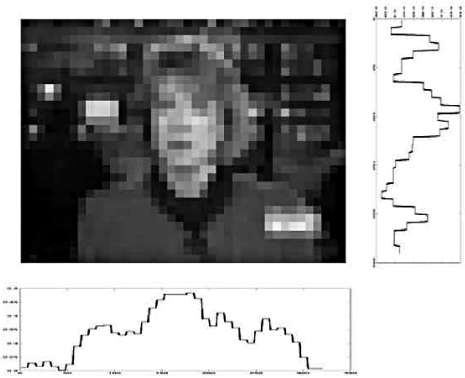
Takto jsou v první úrovní nalezeni kandidáti na obličej, kteří jsou dále zpracováni. V druhé úrovni jsou na kandidáty aplikovány algoritmy na vyrovnání histogramu a detekce hran. Ve třetí

úrovni je existence tváři zjišťována za použití pravidel, která popisují obličejové rysy, kterými jsou oči a ústa. Tento přístup dále rozšířili Kotropoulos a Pitas. V jejich přístupu byly v prvním kroku nalezeny za pomocí projekční metody rysy tváře. Horizontální projekce určovala pozici hlavy. Vertikální projekce lokalizovala ústa, rty, nos a oči. Nechť je $I(x, y)$ hodnota jasu m x n obrazu na pozici (x, y) pak je horizontální projekce definována jako (1) a vertikální projekce je definována jako (2):

$$HI(x) = \sum_{y=1}^{n} I(x, y) \tag{1}$$

$$VI(y) = \sum_{x=1}^{m} I(x, y) \tag{2}$$

Ukázka takového přístupu je na obrázku 11. Dvě lokální minima v horizontální složce korespondují s levou a pravou částí hlavy. Na druhé straně lokální minima ve vertikální složce určují umístěný rtů, nosu a očí. Experimenty ukázaly, že tato metoda není použitelná pro obrazy s mnoha tvářemi a se složitým pozadím.



Obrázek 11: Ukázka vertikální a horizontální složky obrázku [17]

Problém v těchto přístupech nastane v okamžiku, kdy je potřeba přeložit znalosti o lidské tváři do přesně stanovených pravidel. Pokud jsou zvolená pravidla příliš obecná, mohou vzniknout tzv. falešné, klamné neboli nesprávné detekce. V opačném případě s příliš striktními pravidly se může stát, že některé tváře nemusí být vůbec detekovány. Nicméně primární výhoda těchto systémů je v jednoduchosti vytváření pravidel, protože při pohledu na lidskou tvář, je například zřejmá souměrnost očí nebo vzdálenost mezi nimi. Obecně tyto systémy pracují dobře pro lokalizační účely pouze v jednoduchých pozadích a pro detekci tváří, které jsou zobrazeny čelně.

4. Přístupy založené na rysech tváře (Feature - Based)

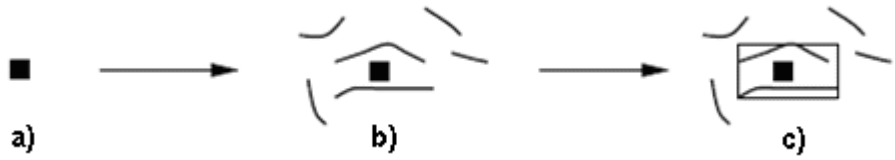
Základ této metody spočívá v nalezení neměnných rysů tváří vhodných pro jejich detekci. Předpokladem pro tuto metodu je zjištěné pozorování, že lidé mohou bez větších potíží lokalizovat tváře, které jsou ovlivněny střídáním světelných podmínek a odlišnými pózami.

To vedlo k myšlence, že musí existovat vlastnosti či rysy lidských tváří, které jsou neměnné vůči těmto proměnlivostem. Četné metody byly navrženy tak, aby první detekovaly obličejové rysy a z nich poté odvozovaly přítomnost tváře. Obličejové rysy jako obočí, oči, ústa, nos a vlasy jsou obvykle extrahovány za pomocí hranových detektorů. Problém u algoritmů založených na této metodě může nastat v okamžiku, kdy jsou obrazové rysy hodně poškozené vlivem například šumu nebo vlivem nevhodného osvětlení. V tom případě jsou hranice určující tvar obličeje oslabeny, a algoritmy pak mohou být nepoužitelné. Existuje několik metod založených na tomto scénáři.

Jednu z nich představili ve své práci Kin Choong Yow a Roberto Cipolla [2]. V prvním kroku je na obraz aplikovaný Gaussův filtr, který slouží pro zjištěný zájmových bodů, které mohou signalizovat možné obličejové rysy (obrázek 12).

Obrázek 12: Model tváře podle Kin Choong Yow a Roberto Cipolla [2]

Ve druhém kroku jsou vyšetřeny hrany kolem zájmových bodů a jsou seskupeny do regionů (obrázek 13).



Obrázek 13: a) Detekce zájmových bodů b) Detekce hran c) Detekce rysu a regionu [2]

Rozměry charakteristických vlastností regionu (délka hran, síla hran) jsou uloženy do vektoru příznaků. Z výcvikových dat obličejových rysů, je poté vypočítán vektor a kovarianční matice pro každý obličejový rys. Například pro obočí je vypočítán vektor $\mu_{obočí}$ a kovarianční matice $\Sigma_{obočí}$, které definují platnou třídu „obočí“. Oblast v obraze je označena jako obličejový rys v případě, že vzdálenosti (záleží na zvolené metrice) mezi odpovídajícími příznaky vektorů jsou

pod určitou hodnotou prahu. Označené rysy jsou dále spojovány do modelu tváře. Nakonec jsou každý obličejový rys a skupiny těchto rysů ohodnoceny bayesovskou sítí, která zajistí snížení počtu nesprávně detekovaných obličejů.

Další přístup představil While Sirohey, který pro segmentaci rysů tváře používal Cannyho detektor hran. Mnoho metod v této kategorii se opírá právě o využití detektorů hran, za pomocí kterých získávají hlavní rysy tváře. Detekce hran je tedy jeden z prvních a důležitých kroků v těchto přístupech. Vzhledem k tomu, že existuje celá řada detektorů hran, vznikala i celá řada detektorů tváří, která se o ně opírala.

Významným rysem, který mohl být dále využit v detektorech, byla barva kůže. Ačkoli existují lidé s různou barvou pleti, vědecké studie ukázaly, že velký rozdíl je spíše v intenzitě než v barevné informaci. Na základě tohoto zjištěný vznikaly mnohé detekční metody, založené na tomto pozorování. Mnohé z nich se snažily normalizovat barevný model RGB, aby mohla být jasová informace filtrována ven. Normalizované barvy byly v těchto přístupech odvozené z původního RGB modelu za pomocí těchto vztahů (3, 4, 5):

$$r = \frac{R}{R+G+B} \tag{3}$$

$$g = \frac{G}{R+G+B} \tag{4}$$

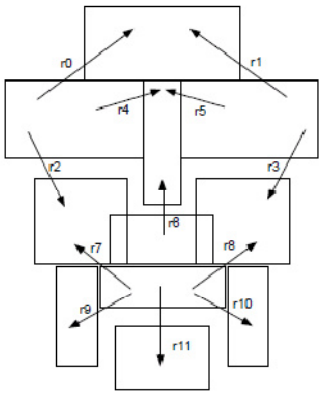
$$b = \frac{B}{R+G+B} \tag{5}$$

Z uvedených vzorců lze odvodit, že $r + b + g = 1$. Normalizované barvy mohou být efektivně reprezentovány jen hodnotami r a g a hodnota b může být dopočítána podle vzorce $b = 1 - r - g$. Za pomocí histogramu založeného na hodnotách r a g bylo dále vypozorováno, že tváře se v něm jeví jako malý shluk. Díky porovnání barevných informací jednotlivých pixelů, s ohledem na hodnoty tváře ve shluku, může být vypočtena pravděpodobnost pixelu patřícího do regionu tváře. Kromě RGB modelů bylo využito i několik alternativních modelů. Mezi takové modely patří HSV, YCrCb, YIQ a další dostupné modely. V mnohých přístupech se také různě rysy tváře kombinovaly, pro lepší dosažení detekce.

5. Přístupy založené na šablonách tváře (Template - Based)

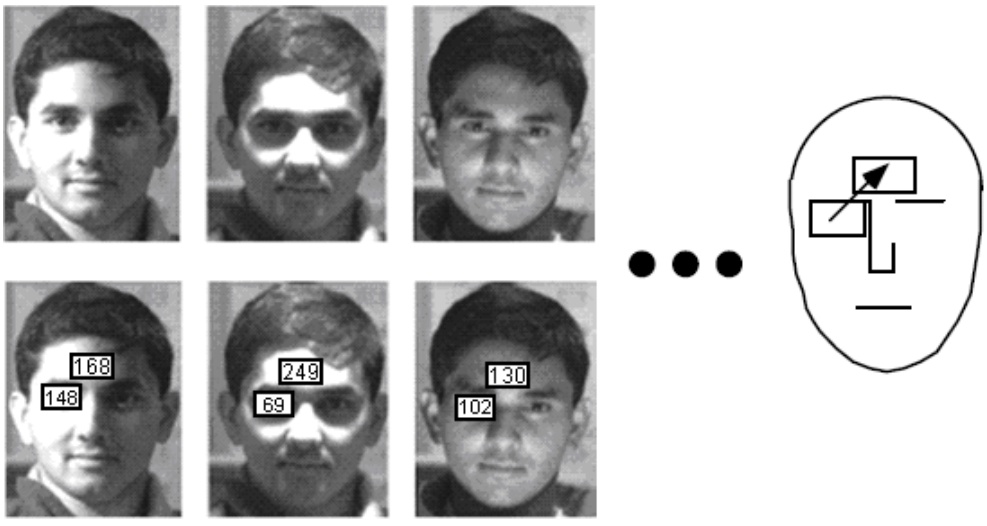
V tomto přístupu je uloženo několik vzorových tváří (šablon) k tomu, aby popisovaly tvář jako celek nebo pro samostatný popis obličejových rysů. Mezi vstupním obrazem a uloženými vzorovými tvářemi je poté vypočtena korelace. Na základě této korelace se rozhodne výsledek detekce.

Jedním ze zástupců této skupiny je systém, který v roce 1996 navrhnul P. Sinha. Jeho algoritmus využívá šablonu tváře, která je složena z několika regionů (obrázek 14).



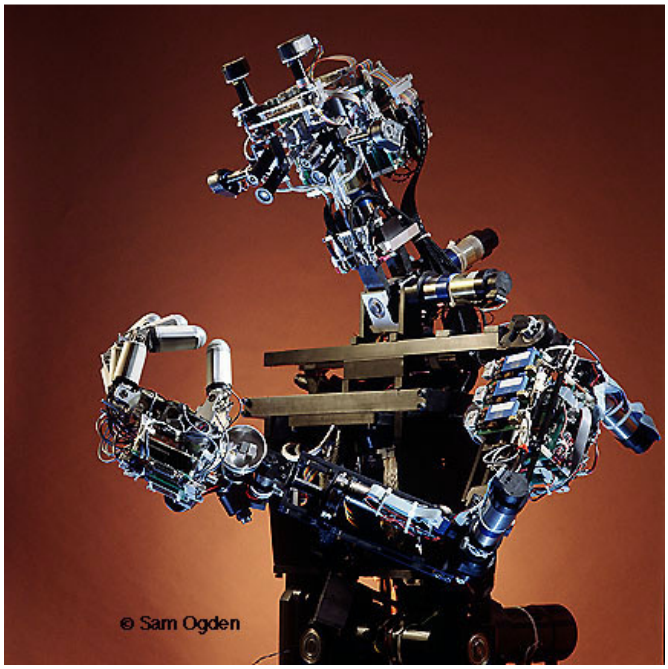
Obrázek 14: Vztahy mezi regiony tvořící společně šablonu tváře [4]

Pro každé cílové umístěný v obraze je vykonáno šablonové srovnání za použití vztahů neboli pravidel mezi jednotlivými regiony. Například průměrný jas v regionu reprezentujícího levé oko bývá menší než jas regionu reprezentujícího čelo, bez ohledu na světelné podmínky (obrázek 15).



Obrázek 15: Praktická ukázka vztahů mezi regiony [4]

Jinými slovy vztah mezi jasem v části levého oka a mezi jasem v oblasti čela je invariantní vůči světelným změnám. Takový přístup vede k vytvoření detekčního systému, který je do jisté míry imunní vůči světelným změnám. V šabloně je vztah přijat, jestliže převod mezi dvěma regiony překračuje daný práh a tvář je lokalizována, jestliže i počet přijatých vztahů překračuje určitý práh. Tento systém byl například aplikován v humanoidním robotovi, vyvíjeném na Massachusetts Institute of Technology (obrázek 16).



Obrázek 16: Humanoidní robot Cog [3]

Existují dvě různé kategorie metod založených na šablonách. První skupinou tvoří tzv. předdefinovaných šablony a druhou tzv. přetvořitelné nebo deformovatelné šablony. Do první skupiny tzv. předdefinovaných šablon spadá i výše popsaný přístup. Výhoda těchto přístupů je v jejich relativně jednoduché realizaci. Nicméně se u těchto metod prokázala nepoužitelnost v okamžiku, kdy jsou v obraze tváře s různými velikostmi nebo s různými úhly natočení. Pro eliminaci těchto neduhů byly vytvořeny tzv. přetvořitelné nebo deformovatelné šablony, které si do jisté míry dokázaly s natočenými nebo velikostně rozdílnými tvářemi v obrazech poradit.

6. Přístupy založené na zpracování obrazu (Image - Based)

Předešlé přístupy mohou mít problém s nepředvídatelností obličejových vzhledů a s podmínkami prostředí. To vedlo k vzniku dalšího přístupu, který by dokonce splňoval i požadavek detekovat tváře v intenzivně znečištěném pozadí a na který by bylo nahlíženo jako na rozpoznávání vzorů (pattern recognition). Taková detekce by využívala strojové učení ze vzorových obrazů k tomu, aby rozpoznala tváře.

Proměnné určené k tomu, aby nesly rozlišující a charakterizující informace o objektu, který má být identifikován se nazývají příznaky. Sada takových příznaků je poté označována jako vektor příznaků. Kategorie, ke které daný objekt náleží, se označuje jako třída. Kolekce příznaků posuzovaného objektu spolu se správnou informací o třídě je nazývána vzorem (pattern). Cílem systémů pro rozpoznávání obrazů je proto správné zařazení objektu do třídy, která koresponduje s daným vektorem příznaků, založeným na nějaké dřívější znalosti získané skrze trénování.

Trénování je procedura, kterou se systémy pro rozpoznání obrazů učí zakreslit vztah mezi příznaky a jejich odpovídajícími třídami. Tento vztah vytváří rozdělující hranice mezi třídami. Jako klasifikátor lze poté označit za zařízení, které dané obrazové prvky zařadí do určité třídy. U detekce obličejů si lze princip představit jako naučení se rozeznávat tváře nebo vše, co tváří být nemá „non-face“, z dané sady kladných a záporných vzorů.

Na tomto přístupu je založeno několik metod. Mezi tyto metody patří detektory založené například na principech Eigenfaces, Fisher's Linear Discriminant, Neural network, support vector machines.

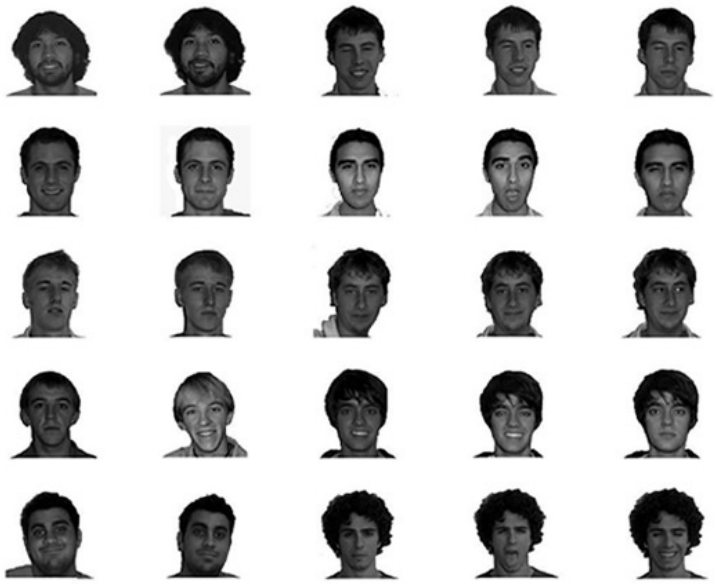
6.1 Eigenfaces

Motivace pro následující metodu může být popsána takto: Úkolem detekce tváří je vybírat vstupní obrazová data do několika tříd. Vstupní signály mohou obsahovat vysokou hladinu tzv. šumu například způsobeného různým osvětlením. Naštěstí vstupní obrazy nejsou úplně náhodné a navzdory jejich rozdílům se v každém z nich vyskytují vzory. Takové vzory, které mohou být vypozorovány ve všech signálech – například přítomnost očí, úst, tváří. Tyto charakteristické rysy jsou nazývány jako eigenface nebo obecně jako hlavní komponenty a mohou být extrahovány z původních obrazových dat prostřednictvím analýzy hlavních komponent – PCA známé také jako Karhunen-Loeve metoda pro redukci dimenze příznakového prostoru. Prostřednictvím PCA může být transformován každý originální obraz výcvikové sady do odpovídajícího eigenfaces. Každý

eigenfaces představuje jen určité rysy obličeje, které mohou být zastoupeny v originálním obraze, některé mohou být zastoupeny více jiné zase méně. Proto každý eigenfaces má mít jistou váhu. Tato váha specifikuje do jaké míry je daný rys (eigenface) přítomný v originálním obraze. Takových rysů může být obecně mnoho, proto je snaha vybrat jen ty nejdůležitější, ze kterých bude možné zpětně za pomocí správných kroků rekonstruovat originální obraz. Navíc opomenutí některých eigenfaces vede ke zlepšení výpočetního výkonu. Algoritmus detekce tváří může být popsán následovně:

Originální obrazy výcvikového sady jsou transformovány do množiny eigenfaces E. Dále se stanoví váhy pro každý obraz výcvikové sady a uloží se do množiny W. Při pozorování neznámého obrazu X se určí jeho váhy a uloží se do vektoru Wx. Následně se Wx porovná s váhami obrazů, o kterých jistě víme, že jsou tváře. To může být realizováno tak, že na každý váhový vektor budeme pohlížet jako na bod v prostoru a budeme počítat průměrné vzdálenosti mezi nimi. Pokud zjištěná průměrná vzdálenost překračuje nějakou prahovou hodnotu neznámy obraz X není považován za tvář v opačném případě je X označen jako obličej.

Uvažujme výcvikovou sadu obrázků tváří (obrázek 17), na kterou bude nahlíženo jako na sloupcový vektor n = WxH, kde W, H jsou rozměry obrázků. Délka takového vektoru určuje dimenzi příznakového prostoru. Za pomocí PCA lze poté nalézt takovou reprezentaci, ve které bude obrázek vyjádřen vektorem délky m << n (odtud „redukce příznakového prostoru“).



Obrázek 17: Ukázka výcvikové sady [8]

Postup výpočtu eigenfaces za použití PCA:

1. Vstupem je sada vektorizovaných obrázků tváří $\Gamma_1, \Gamma_2, \ldots, \Gamma_m$ – každý obrázek si lze představit jako vektor

2. Je třeba vypočítat průměrný obraz (obrázek 18) této sady podle vztahu (6):

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{6}$$

Obrázek 18: Průměrný obraz tváře [8]

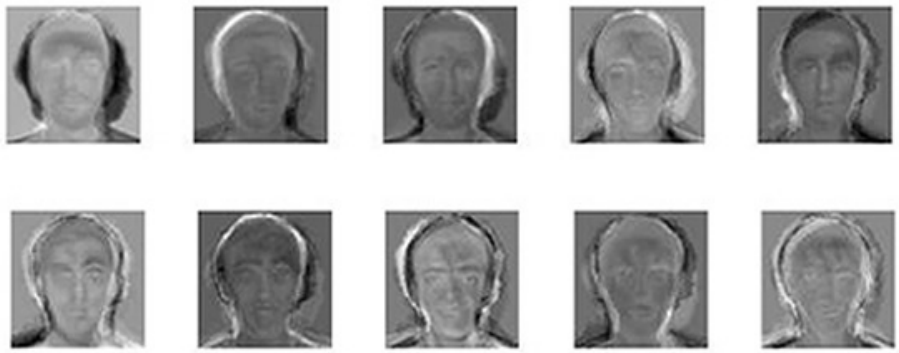
3. Dalším krokem je nalezení rozdílu mezi vstupním obrazem a obrazem průměrným (7)

$$\Phi_i = \Gamma_i - \Psi \tag{7}$$

4. Takto získané vektory se použijí jako vstup do analýzy hlavních komponent, která hledá množinu n ortonormálních vektorů u_i, které nejlépe popisují distribuci vstupních dat. Vektory u_k a skaláry λ_k jsou vlastní vektory neboli hlavní komponenty resp. Vlastní čísla kovarianční matice (8):

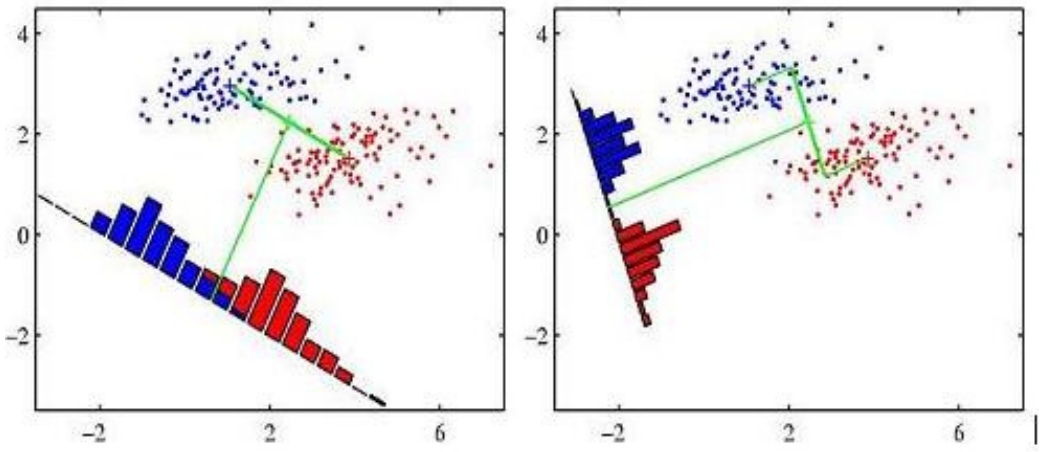
$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \tag{8}$$

5. Z množiny M vlastních vektorů (eigenfaces) je vybrána podmnožina M', podle velikosti jejich vlastních čísel. Eigenfaces s nízkými hodnotami vlastních čísel mohou být vynechány, protože nejméně charakterizují část příznačných obličejových rysů. V této části výcviková fáze algoritmu končí. Na obrázku 19. je ukázka deseti eigenfaces.



Obrázek 19. Ukázka eigenfaces [8]

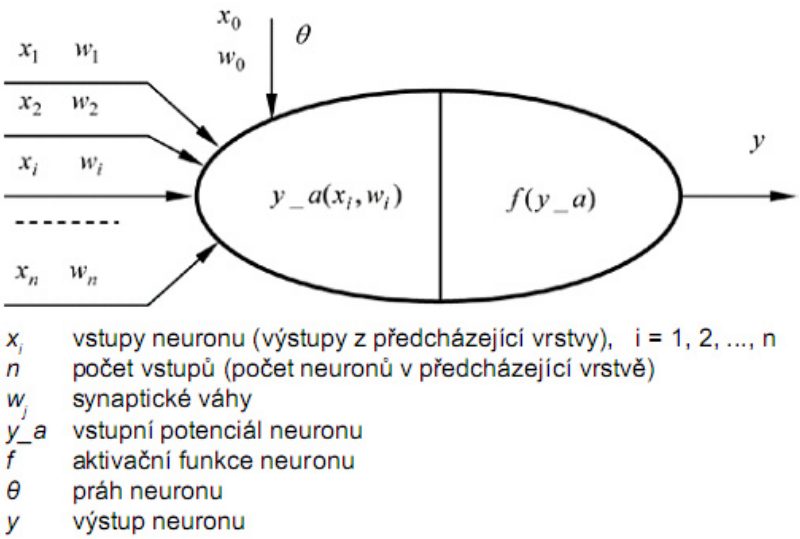
Další silný nástroj pro redukci prostoru a extrakci rysů je LDA (*Linear Discriminant Analysis*). LDA využívá třídní informace a nalézá soustavu vektorů, které maximalizují rozptyl mezi třídami a minimalizuje rozptyl uvnitř tříd (obrázek 20). Jinými slovy LDA se snaží nalézt nejlepší projekci mezi třídami. Například v článku [9] autoři srovnávají výsledky metod PCA a LDA při rozpoznávání tváří.



Obrázek 20: Minimalizace variability uvnitř tříd a maximalizace variability mezi třídami, kde byla použita Fisherova diskriminační funkce. Špatná třídní rozlišení (levá strana obrázku) vs. dobré třídní rozlišení (pravá strana obrázku). [10]

6.2 Neuronové sítě

Umělé neuronové sítě si berou za vzor z pohledu funkcionality biologické neuronové sítě. Základem modelu neuronových sítí je tzv. neuron (obrázek 21).



Obrázek 21: Schéma umělého neuronu [5]

Každý umělý neuron obsahuje konečný počet vstupů x_n a jediný výstup y. Tento jediný výstup je samozřejmě možno rozmnožit do potřebného počtu kopií – vstupů do následných neuronů. V každém neuronu (popisovaný základní model nevyjímaje) se vstupní hodnoty transformují na výstup pomocí minimálně dvou výpočetních procedur. Konkrétně se jedná o výpočet vstupního potenciálu y_a a o tzv. aktivační funkci f. [5]

Neuron může být v sítích propojen různými způsoby s dalšími neurony a společně mohou tvořit neuronovou síť. Takové propojení je označováno jako architektura, topologie nebo struktura sítě, kde jsou jednotlivé neurony uspořádány do daných vrstev. Vrstvy se poté dělí podle polohy na vrstvy vstupní, skryté, výstupní a jejich název přebírají i neurony v nich umístěné. Podle informace, která se šíří neuronovou sítí se rozlišují dva základní typy sítí:

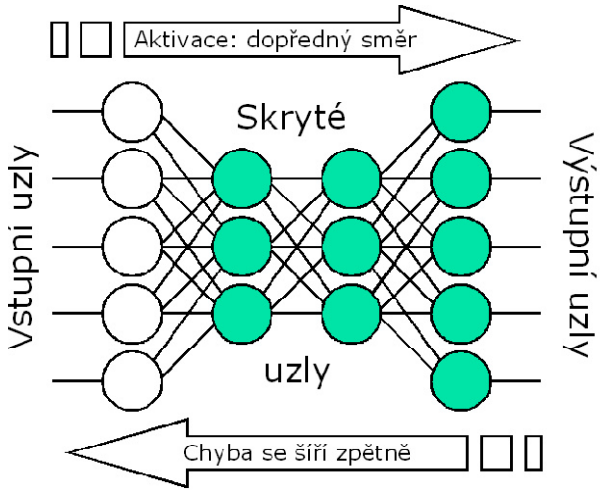
- dopředné sítě – informace se v síti šíří jedním směrem od vstupu na výstup sítě
- rekurentní sítě – v síti existují mezi neurony a vrstvami zpětné vazby, které umožňují tok informace z libovolného výstupního bodu zpět na vstup sítě nebo vrstvy

Jednotlivé parametry neuronové sítě mohou být různě nastavovány. Proces, který tuto operaci provádí je nazýván učení nebo trénování. Většinou se jedná o nastavování prahových a váhových hodnot. Ty se poté nastavují v závislosti na řešeném problému.

Teorie učení UNS je postavená kolem Hebbova zákona, publikovaného ve čtyřicátých letech, který je inspirován způsobem učení mozku: "Pokud jsou dva neurony v jednom okamžiku aktivní (jeden vybudil druhý), zesil jejich vazbu (příslušná úprava vah), v opačném případě ji zeslab." Tento zákon implementuje v obměnách většina učících algoritmů. Naučení sítě je tedy uloženo ve vahách spojů mezi neurony, samotná topologie se při učení nemění (lze realizovat pouze přerušení spojení neuronů nastavením příslušné váhy na 0). U učení rozlišujeme dva druhy:

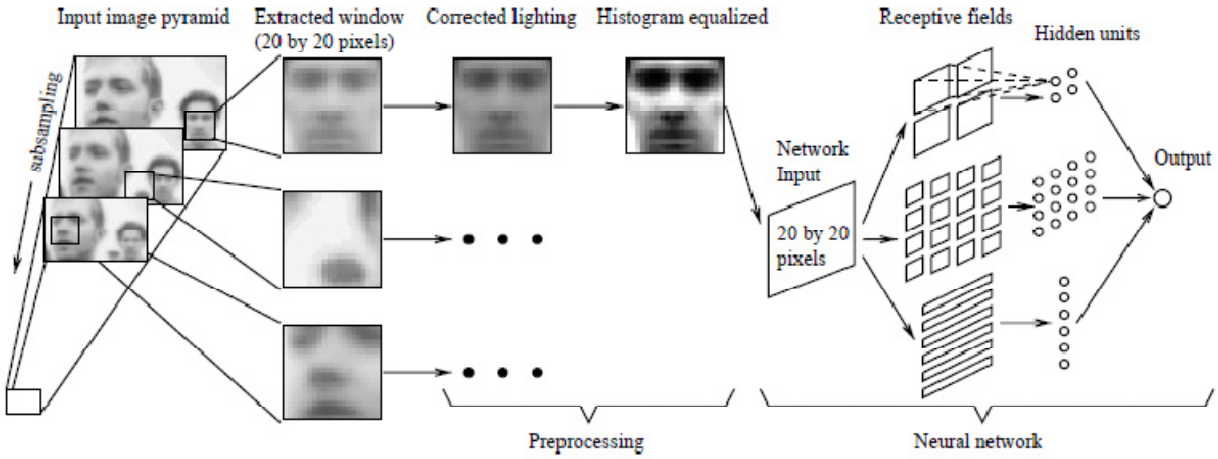
učení s učitelem a bez učitele. V prvním případě máme k dispozici páry vstupní a k němu příslušející výstupní vektor z reálně naměřených dat (tzv. trénovací množina). Algoritmus porovná požadovaný výstup se skutečným výstupem sítě a upraví váhy spojů tak, aby se výstup přiblížil požadované hodnotě. To provede s každým prvkem trénovací množiny. Při učení bez učitele má síť k dispozici pouze vstupní data a hledá v nich shluky - podobná data (shluková analýza). Algoritmus pak posouvá jednotlivé neurony (reprezentované vektorem vah) ve stavovém prostoru. Proces aplikace naučených poznatků se nazývá vybavování. S učením je také spojen jeden z problémů UNS, konkrétně přeučení - síť sice perfektně odpovídá na vstupy z trénovací množiny, ale není schopna generalizace a odpovědi na reálná data jsou nesmyslné. [6]

V praxi se využívá celá řada různých topologií UNS. Mohou to být například: Hopfieldova síť, Hammingova síť, vícevrstvý perceptron (obrázek 22).



Obrázek 22: Vícevrstvý perceptron. Jako učící algoritmus používá zpětné šíření chyby (backpropagation), kdy se po porovnávání skutečného výstupu s očekávaným upravují nejdříve a nejvíce váhy v poslední vrstvě, pak méně v předposlední atd.. [6]

Neuronové sítě mohou být použity pro vytvoření systémů, které budou schopny klasifikovat údaje do jednotlivých tříd. Detekci tváří za použití neuronových sítí představil Henry A. Rowley [7]. Jeho systém přijímá na vstupu region pixelů o velikosti 20x20 a generuje výstup v rozsahu 1 až -1 v závislosti na přítomnosti nebo nepřítomnosti tváře. Pro zjištění tváře ve vstupním obraze je filtrem postupně procházen celý vstupní obraz.



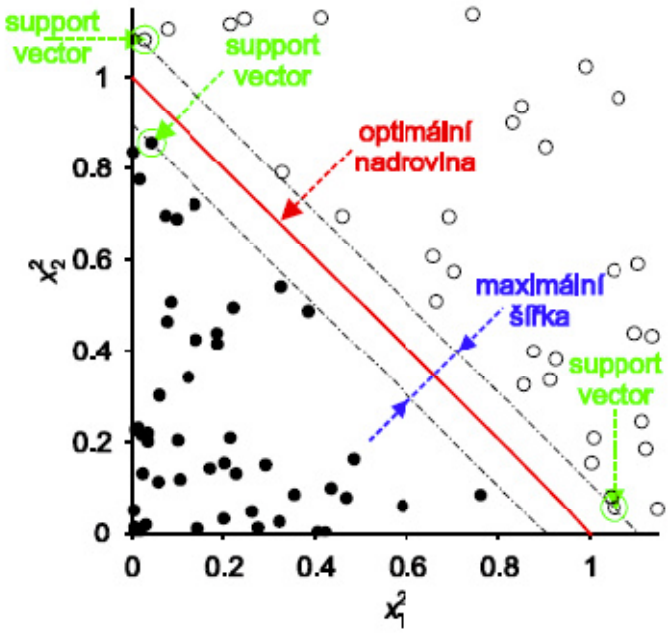
Obrázek 23: Proces použitý při detekci tváří [7]

V další fázi je nutné redukovat výkyvy způsobené osvětlením, kamerovými rozdíly atd. Pro zlepšení jasu a kontrastu je v této fázi na obrazy aplikován algoritmus pro vyrovnání histogramu. Po předzpracování jsou tyto obrazy vpuštěny do samotné neuronové sítě a na základě jejího natrénování síť rozhoduje, zda okno osahuje tvář či nikoliv. Obrázek 23 znázorňuje celý postup při detekci za použití neuronové sítě podle [7].

Metody založené na využití neuronových sítí dosahují dobré detekční výsledky, ale na druhou stranu kvalita detekce je velice závislá na výcvikových sadách a na ladění neuronových sítí, které mohou mít obvykle mnoho parametrů.

6.3 Support vector machines

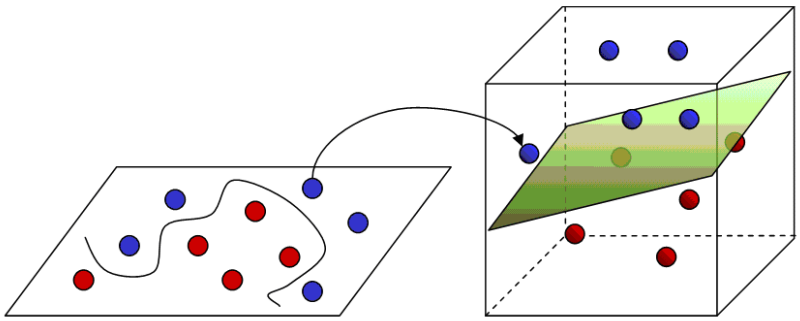
Další možností je využití relativně mladé metody založené na tzv. jádrových algoritmech (kernel machines) s využitím podpůrných vektorů (support vector machines). Metodu SVM představil V. Vapnik. Za pomoci této metody lze najít optimální oddělovač, který bude maximalizovat hranice mezi různými třídami. Jinými slovy se metoda snaží nalézt optimální prostor mezi příklady jedné třídy a příklady třídy druhé (obrázek 23).



Obrázek 24: Optimální oddělovací hranice [11]

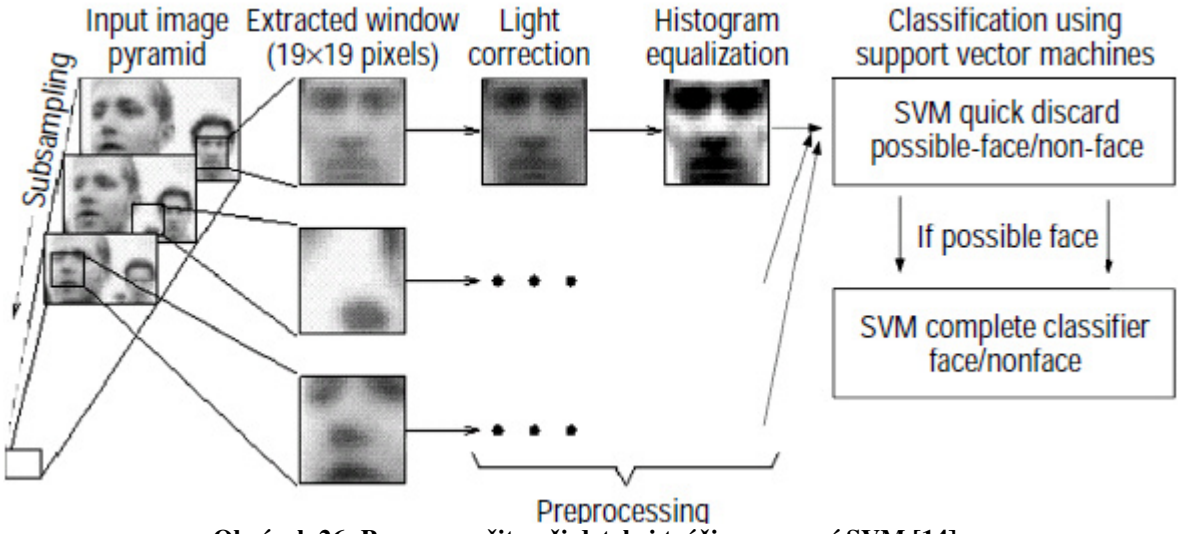
Body, které se nacházejí nejblíže oddělovače, se nazývají podpůrné vektory (support vector) a jsou počítány za pomocí kvadratického programování. K takto nalezené hranici za pomocí podpůrných vektorů je zapotřebí mnohem méně trénovacích příkladů, než bývá zvykem u jiných metod. Ve většině případů se pro výpočet optimální hranice používají všechny příklady z trénovací sady. Naproti tomu u metody SVM jsou zapotřebí pro samotný výpočet jen ty body, které se určí jako významné tj. nejvíce informativní pro určení oddělovače.

Kromě nalezení lineární hranice umožňuje metoda SVM reprezentovat i vysoce nelineární funkce a to tak, že převede původní vstup do vysoce dimenzionálního prostoru, ve kterém již lze od sebe třídy lineárně rozdělit (obrázek 25).



Obrázek 25: Ukázka oddělení dvou tříd lineárním oddělovačem za pomocí přidání dalšího rozměru [13]

Detekci tváří za pomocí SVM představil Osuma[12]. Počáteční předzpracování obrazu je obdobné jako u detekce neuronových sítí v předchozí kapitole, jen na rozdíl od neuronových sítí je zde použit SVM. Proces podle [14] je zobrazen na obrázku 26.



Obrázek 26: Proces použity při detekci tváři za pomocí SVM [14]

6.4 Hidden Markov Model

Dalším nástrojem pro trénování a rozpoznávání jsou tzv. skryté Markovovy modely. Skrytý Markovův model si lze představit jako pravděpodobnostní automat, který v každém diskrétním časovém okamžiku vygeneruje tzv. observaci. Observace může být v závislosti na aplikaci jak jedno reálné (resp. celé) číslo, tak i vektor. Parametry Markovova modelu jsou:

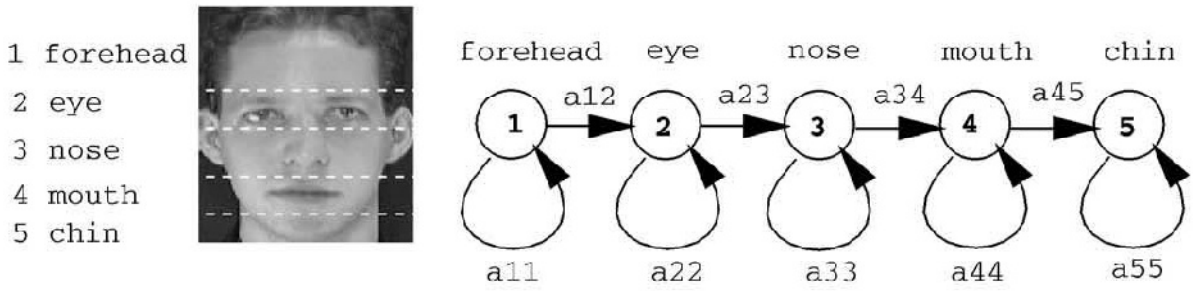
• pravděpodobnosti přechodů mezi jeho stavy značené *aij* (pravděpodobnost přechodu z *i*-tého do *j*-tého stavu)

• tzv. *emisní pravděpodobnosti* značené *bj(o),* které vyjadřují pravděpodobnost, že *j*-tý stav vygeneroval (emitoval) observaci *o*. [15]

Před použitím HMM je nutné trénováním získat tyto parametry modelu. K tomu účelu je možné použít Baum-Welchova nebo Viterbiho algoritmů.

V oblasti detekce nebo rozpoznávání tváří může být tvář rozdělena do několika regionů (čelo, oči, nos, ústa, a brada). Mějme dán proces rozpoznání tváře, ve kterém jsou tyto regiony pozorovány v daném pořadí. Pomocí využití tohoto přístupu založeného na Markovových modelech, mohou být regiony spojeny se stavy skrytého Markovova modelu. Metody založené na

HMM nejčastěji pracují se vzorem tváře jako se sekvencí vektorů observací, kde každý vektor je proužek pixelů. Během trénování a testování je obraz snímán obvykle shora dolů a hranice mezi řezy pixelů jsou reprezentované pravděpodobnostními přechody mezi stavy (obrázek 27), kde jsou obrazová data modelována za pomocí Gaussova rozdělení. Jestliže je získaná pravděpodobnost každého bloku pixelů nad určitou prahovou hodnotou, je v obraze detekován obličej. Takový přístup představil Samaria and Young.



Obrázek 27: Pět možných stavů HMM [17]

Metody založené na těchto přístupech tedy obecně využívají algoritmy z oblasti strojového učení. Tyto metody se také vyznačují potřebou mít k dispozici sadu výcvikových dat a to jak pozitivních tak negativních. Systémy založené na tomto přístupy se zdají být přesné, rychlé a dají se využívat v mnoha prostředích. Tedy i tam, kde se tváře vyskytují různě natočené a v různých velikostech. Podle experimentů publikovaných například v [17] tyto metody vykazují velmi dobré výsledky. Na druhé straně jejich výcviková fáze může být výpočetně i časově velmi náročná.

V poslední době se také hodně hovoří o detektoru Viola-Jones, který se podle různých publikací a experimentů jeví jako velmi schopný. Tento detektor byl v této práci vybrán pro praktickou demonstraci a bude podrobněji popsán v následující kapitole.

7. Viola-Jones face detector

Implementovaný detektor, který byl vybrán do této práce, za pomocí knihovny openCV je založen na metodě, kterou publikovali Paul Viola a Michael Jones v roce 2001. Jedná se o relativně novou metodu, která kombinuje tradiční přístupy založené na geometrických rysech tváře spolu s metodami založenými na zpracování obrazu. Geometrických ve smyslu, že využívá hlavní rysy lidských tváří, především oči, ústa a nos. Na druhé straně využívá obrazově založené přístupy, kdy užívá statistické učení s použitím vyplývající skupiny dat, potřebné k budování modelu tváře. Rychlost odhalení tváře je dána jednoduchostí vybraných příznaků a dobrá objasněnost tváří je získána použitím fundamentálního posilujícího algoritmu AdaBoost. Funkčnost detektoru spočívá ve skenování obrazu posouvajícím oknem tzv. výřezy. Každý výřez je testován klasifikátorem v několika stupních (představa vodopádu, kaskády). Pokud výřez není zřetelně tváří, je odmítnut a pokračuje dalšími kroky v kaskádě. Potíž je v tom najít dost jednoduché příznaky, aby umožnily rychlé klasifikace, ale zároveň musí rozlišovat tváře od falešných detekcí. V další sekci budou takové příznaky představeny.

Pro zlepšení výpočetní doby těchto příznaků Viola and Jones využívají obrazovou reprezentaci nazvanou jako Integral Image, tedy integrální obraz. Tato reprezentace bude podrobněji představena v dalších kapitolách. V okamžiku, kdy je k dispozici sada příznaků, Viola nad Jones využívají již výše zmíněný algoritmus AdaBosst pro výběr jen malé sady těchto příznaků a ke konstrukci konečného klasifikátoru. Celý proces výběru příznaků a sestrojení klasifikátoru bude také vysvětlen v dalších sekcích této kapitoly. Jedna z částí bude věnována i konstrukci a významu kaskádové detekce.

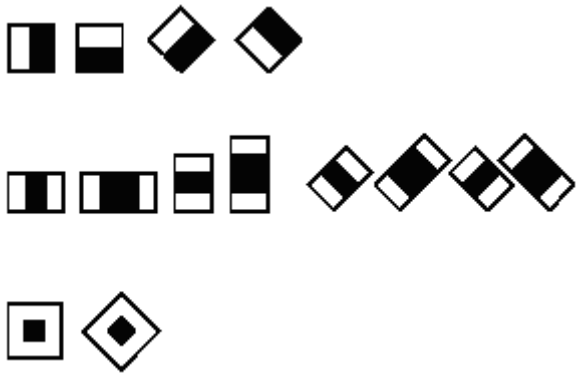
7.1 Příznaky

Jak již bylo naznačeno v kapitole 6, za pomocí příznaků mohou být popsány předměty nebo jevy, které je snaha klasifikovat neboli rozpoznávat do jednotlivých tříd. Ve třídě by se poté měli nacházet předměty, mající určité společné vlastnosti, tedy příznaky. V našem případě se jedná o skupinu dvou tříd. První skupinou jsou tváře v obrazech a druhou skupinou je vše, co tváří být nemá. V oblasti rozpoznávání tváře se v poslední době hovoří o dvou přístupech při realizaci vlastností daných objektů. Jedná se o skupinu systémů založených právě na příznacích a o další skupinu systémů založených na pixelech.

Metodu modelování tváře za použití pixelů a algoritmu AdaBoost představil ve své práci Vladimir Pavlovič [23]. Ten ve svém přístupu používá klasifikátory založené na hodnotách

jednotlivých pixelů. Algoritmus AdaBoost poté využívá pro výběr skupiny pixelů, které reprezentují nejlepší strukturu tváře. I když se tato metoda jeví jako účinná, výpočetní čas může být vysoký, vzhledem k tomu, že je třeba prozkoumat každý klasifikátor založený právě na jednom pixelu. Viola a Jones se rozhodli jít jinou cestou, kvůli zlepšení výpočetní náročnosti.

Identifikační procedura, kterou metoda Viola and Jones používá, klasifikuje obrazy podle hodnot jednoduchých příznaků, které jsou založeny na Haarových vlnkách. Možné použitelné příznaky jsou zobrazeny na obrázku 28. Hodnota příznaku je určena jako rozdíl mezi sumou obrazových pixelů pod černou a bílou částí regionů. Příznaky zobrazeny na obrázku 28 nejsou jedinými příznaky, které je možné použít. V Mnoha vědeckých publikacích bylo představeno několik různých prototypů těchto příznaků. Na obrázku 28 jsou vyobrazeny jen některé z nich. Na prvním řádku jsou tvarově nejjednodušší příznaky, na řádku třetím je poté ukázka již složitějších prototypů těchto příznaků.



Obrázek 28: Sada Haarových příznaků

Příznak je definován svým tvarem a velikostí. Od toho se také odvíjí počet jednotlivých příznaků. Za předpokladu, že základní rozlišení detektoru je okno o velikosti 19x19 pixelů může být v takovém okně až několik tisíc různých příznaků. Příznak tedy nese informaci o tom, jestli se v dané oblasti vyskytuje tvář či nikoliv. Výběr a nároky na jednotlivé příznaky budou popsány v kapitole o algoritmu AdaBoost.

K tomu, aby byl detektor úspěšný je potřeba nalézt kompromis mezi jeho rychlostí a přesností. Za pomoci obrazové reprezentace nazvané Integral Image, Viola popisuje způsob pro rychlé vyhodnocení, které se prokázalo jako efektivní způsob pro urychlení klasifikace systému.

7.2 Integrální obraz

Prvním krokem detektoru je tedy změnit vstupní obraz na obraz integrální podle (9):

$$I(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{9}$$

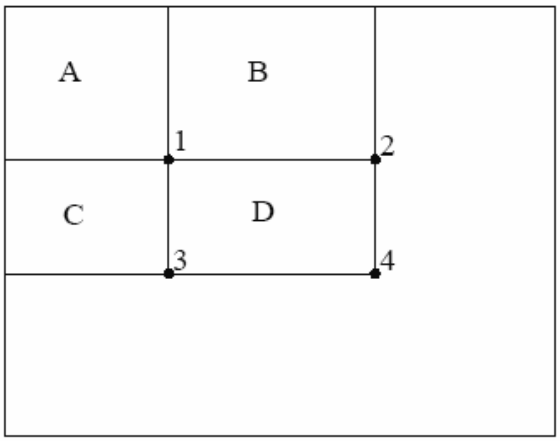
Na vzorci je Integrální obraz označený jako *I(x, y)*, na poloze *(x, y)* obsahuje sumu jasových hodnot pixelů nad a vlevo od souřadnic *(x, y)*, kde *i* je vstupní obraz. Praktická ukázka je na obrázku 29.

1	1	1
1	1	1
1	1	1

1	2	3
2	4	6
3	6	9

Obrázek 29: Na levé straně vstupní obraz na straně pravé obraz integrální

Tento přístup umožní výpočet libovolného pravoúhelníku za použití pouze čtyř hodnot. Tyto hodnoty jsou pixely v integrálním obraze, které se shodují s rohy ve vstupním obraze. Ukázka je na obrázku 30.



Obrázek 30: Hodnota integrálního obrazu na pozici 1 je suma pixelů v pravoúhelníku A. Hodnota pozici 2 je A+B, na poloze 3 je A+B+C a na pozici 4 je A+B+C+D. Suma uvnitř D poté může být vypočítána jako 4+1 – (2+3).

Díky integrální reprezentaci obrazu lze jednotlivý příznak vypočítat v každém umístění obrazu jen za použití pár operací. V okamžiku kdy jsou k dispozici vypočítané příznaky je potřeba z nich vybrat jen určitou část, která nejlépe popisuje daný objekt v našem případě obličej. K tomu účelu systém Viola-Jones používá algoritmus AdaBoost popsaný v následující kapitole.

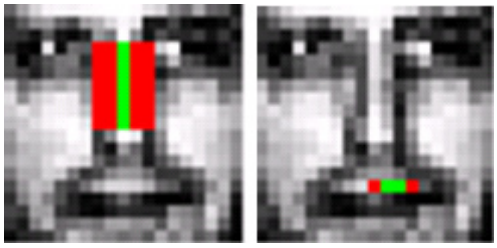
7.3 AdaBoost obecně

Algoritmus AdaBoost představili ve své práci v roce 1996 pánové Freund a Schapire[25]. AdaBoost (adaptive boosting) spadá do kategorie algoritmů, které jsou označovány jako boosting metody. Hlavní myšlenka principu boosting spočívá v kombinaci několika klasifikátorů, které mají klasifikační přesnost lepší než 50%. V okamžiku přidávání dalších klasifikátorů se stejnými požadavky na klasifikační přesnost (tedy klasifikační přesnost lepší než 50%) do skupiny klasifikátorů lze dosáhnout u této skupiny prakticky libovolně vysokou klasifikační přesnost. V takovém případě se hovoří o zesílení (boosted) klasifikace. AdaBoost ve svém názvu nese přívlastek adaptivní (adaptive) v tom smyslu, že se dokáže zaměřit na těžce klasifikované příklady. Algoritmus pracuje v iteracích, kdy v každé iteraci zvedne váhu těch příkladů, které byly chybně klasifikovány. Například příklady nacházející se blízko oddělující hranice dostanou v prvních iteracích větší váhové ohodnocení, protože jejich zařazení do třídy je většinou obtížnější. Dále bude význam metody přestaven při využití v rozpoznávání a v detektoru Viola-Jones.

7.4 AdaBoost v rozpoznávání

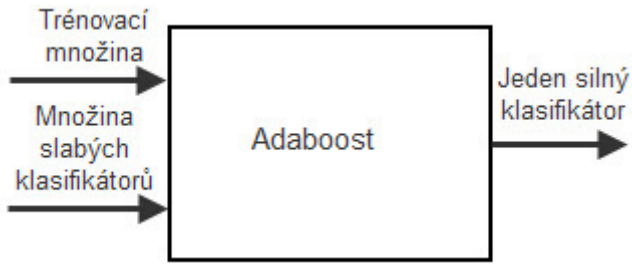
Z předchozích kapitol jsou již známy použité příznaky a obecný princip metody boosting. Nyní bude následovat jejich využití právě v algoritmu AdaBoost. Pro lepší pochopení budou některé souvislosti připomenuty a objasněny na praktických příkladech.

Systém Viola a Jones používá pro trénování klasifikátoru modifikaci diskrétní verze algoritmu AdaBoost (základní schéma na obrázku 32), která se jeví jako efektní při hledání malých počtů dobrých příznaků (obrázek 31).



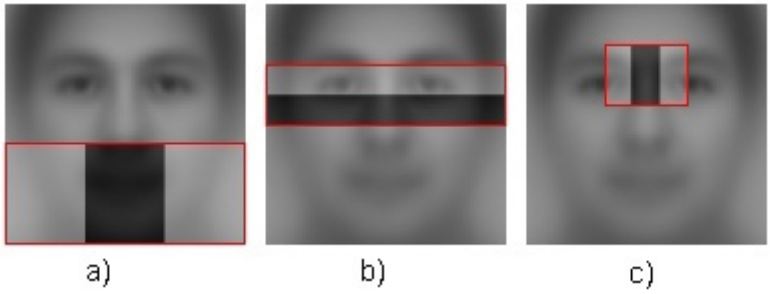
Obrázek 31: Významný (levý) vs. nevýznamný (pravý) příznak [26]

Dále je algoritmus použit při kombinaci několika slabých klasifikátorů do jednoho silného. Jinými slovy princip metody boosting spočívá v kombinování jednoduchých klasifikátorů, které se nazývají slabí žáci. Slabí žáci proto, že se od nich neočekává nejlepší klasifikace do tříd, ale stačí, aby pracovali o něco lépe než náhodná funkce.



Obrázek 32: Blokové schéma AdaBoost

K tomu se využívá spojení, kdy je jednomu slabému klasifikátoru přiřazen právě jeden příznak. V takovém případě je možné použít algoritmu AdaBoost pro výběr příznaku, který poskytuje nejlepší separaci mezi kladnými a zápornými příklady (ukázka výběru takových příznaků na obrázku 33).



Obrázek 33: Příklad trojice vybraných příznaků pomocí AdaBoost. První příznak a) měří rozdíl intenzity mezi regionem úst a regionem čelistí. Druhý příznak b) měří rozdíl intenzity mezi regionem lícní částí tváře a regionem očí, které bývají v mnoha případech tmavější než jiné regiony. Třetí možný příznak c) porovnává intenzity regionu nosního můstku a regionu očí. [22]

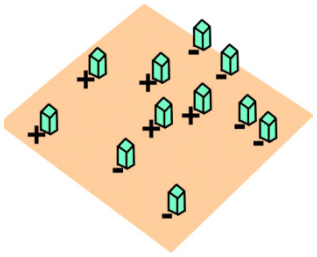
Jak již bylo zmíněno, odezva příznaku je tedy rozdíl sumy pixelů pod černou a bílou částí regionů. Za předpokladu, že tato odpověď umožní odlišit kladné a záporné příklady, lze slabý klasifikátor popsat vztahem (10):

$$h_j(x) = \begin{cases} 1 & jestliže\ p_j f_j(x) > p_j \theta_j \\ 0 & jinak \end{cases} \qquad (10)$$

Kde slabý klasifikátor $hj(x)$ je sestaven z příznaku fj, polarity pj a prahové hodnoty θj, která rozhodne, zda vážený příklad x bude klasifikován jako tvář či nikoli. Vážený ve smyslu, že všechny příklady jsou opětovně váženy v každém stupni algoritmu. Pseudokód algoritmu

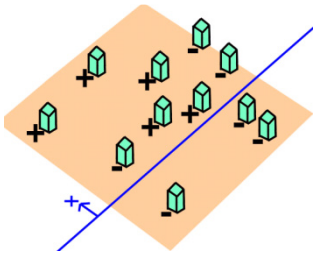
AdaBoost by v tomto případě mohl vypadat následovně:

1. AdaBoost začíná s jednou distribucí vah přes výcvikové příklady. Váhy říkají algoritmu význam příkladu. Na obrázcích 34-37 představuje každý zelený hranol jeden příklad. Pod každým hranolem je zobrazena jeho příslušnost ke třídě. Na obrázcích jsou třídy dvě {-, x}.



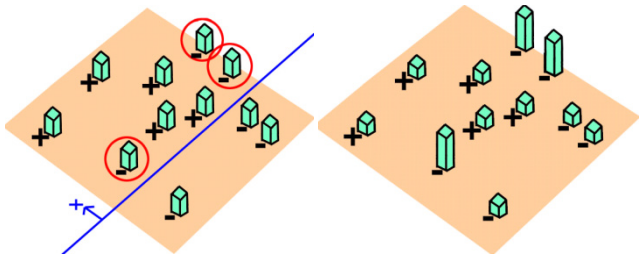
Obrázek 34: Jednotná distribuce vah [26]

2. Vyber nejlepší slabý klasifikátor vzhledem k daným vahám příkladů. Na obrázcích 35-38 jsou vybrané slabé klasifikátory znázorněny pomocí modrého oddělovače.



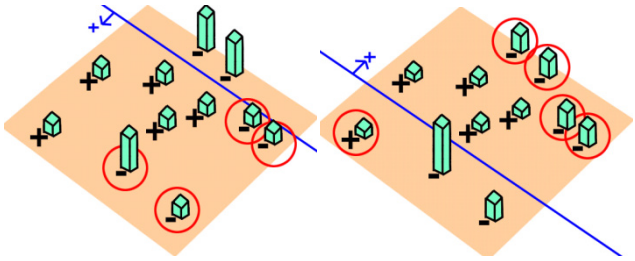
Obrázek 35: Výběr klasifikátoru vzhledem k vahám [26]

3. Zvyš váhy chybně klasifikovaných příkladů a sniž váhy dobře klasifikovaných příkladů. Chybně klasifikované příklady jsou na obrázcích 36, 37 označeny červeně. U špatně klasifikovaných příkladů (hranolů) jsou na obrázcích 36,37 zvýšeny jejich velikosti. Naproti tomu u správně klasifikovaných příkladů (hranolů) na obrázcích 36,37 jsou jejich velikosti sníženy.



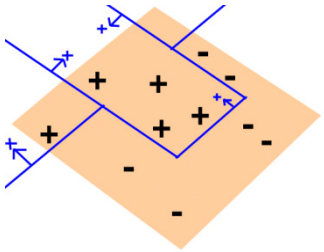
Obrázek 36: Zvýšení vah chybně klasifikovaných příkladů [26]

4. Opakuj kroky 1-3 (obrázek 37). První opakování je na obrázku 37 zobrazeno na levé straně, druhé opakování je zobrazeno na straně pravé. Po těchto opakováních cyklus ukonči.



Obrázek 37: Opakování předchozích kroků [26]

5. Na konci udělej lineární kombinaci slabých klasifikátorů získaných v každé iteraci. Na obrázku 38 je vidět vytvořená výsledná hranice modrým oddělovačem.



Obrázek 38: Kombinace klasifikátorů [26]

Kompletní algoritmus podle Viola-Jones:

- Mějme trénovací množinu obrazů ($x1$, $y1$), . . . , (xn , yn) kde yi = 0,1 značí negativní respektive pozitivní příklady.
- Inicializuj váhy $w1,i = 1/2m$, $1/2l$ pro yi = 0,1 kde m je počet negativních a l je počet pozitivních příkladů
- Pro t = 1,..., T:
  1. Normalizace vah (11)

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{11}$$

  tak aby w_t bylo pravděpodobnostní rozložení
  2. Trénuj klasifikátor tak aby chyba ε_j (12) byla nejmenší vzhledem k distribuci w_t

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right| \tag{12}$$

3. Vyber klasifikátor h_t s nejmenší chybou ε_t
4. Aktualizuj váhy podle (13)

$$w_{t+1} = w_{t,i} \beta_t^{1-e_i} \tag{13}$$

kde $e_i = 0$ pokud je příklad správně klasifikovaný, jinak $e_i = 1$, a

$$\beta_t = \frac{e_t}{1 - e_t} \tag{14}$$

- Finální silný klasifikátor je (15):

$$h(x) = \begin{cases} 1 \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \ jinak \end{cases} \tag{15}$$

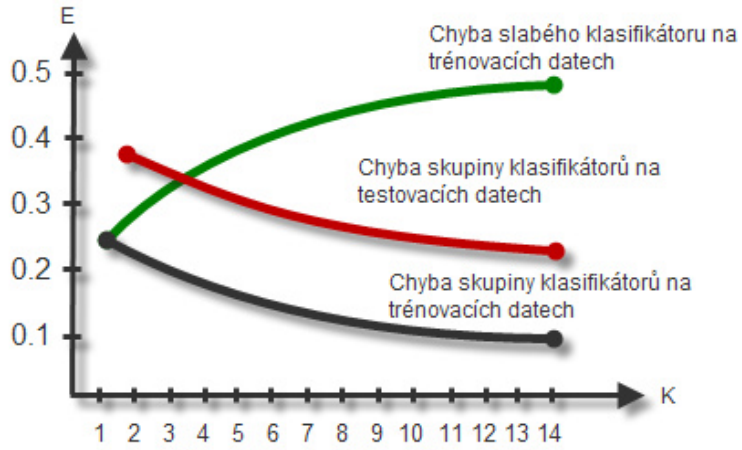
Kde

$$\alpha_t = \log \frac{1}{\beta_t} \tag{16}$$

AdaBoost na vstupu bere výcvikovou sadu $S = (x_1; y_1), ..., (x_m; y_m)$, kde každá instance příkladu x_i je z prostoru příkladů X, a každé ohodnocení y_i spadá do prostoru ohodnocení Y. Jedná se o binární případ kde Y = {0, 1}. Pozitivní příklady jsou označeny hodnotou 1, negativní příklady hodnotou 0. Algoritmus tedy funguje iterativním způsobem a v každé iteraci $t = 1, ...T$, mění důležitost vstupních příkladů. Ze všech možných slabých klasifikátorů vybere ty, které vykazují nejlepší výsledky. Na konci algoritmu je poté získán lineární kombinací slabých klasifikátorů jeden silný klasifikátor.

V závislosti na vybraných klasifikátorech algoritmus AdaBoost snižuje exponenciálně trénovací chybu. Jak již bylo zmíněno AdaBoost se tedy zaměřuje na těžce klasifikované případy. Kvůli tomu dosahuje obecně každý další přidaný klasifikátor na trénovacích datech větší chybu než kterýkoliv z předchozích (křivka zelenou barvou na obrázku 39 - *Chyba slabého klasifikátoru na trénovacích datech*). Ale rozhodnutí skupiny klasifikátorů zaručuje pokles chyby na trénovacích datech do okamžiku, kdy každý z klasifikátorů ve skupině dává lepší než náhodný výsledek. V takovém případě dochází k poklesu i na datech testovacích (křivka červenou barvou na obrázku

39). Obecně, ale není vhodné zvyšovat počet klasifikátorů ve skupině, kvůli možnému vzniku tzv. přetrénování. Tedy ke ztrátě schopnosti generalizace. Nicméně podle obecných experimentů s algoritmem k tomu nedochází tak často a algoritmus AdaBoost se právě vyznačuje dobrou schopností generalizace. V dnešní době existuje několik variant tohoto algoritmu. Může se jednat o varianty Real AdaBoost, Gentle AdaBoost, LogiBoost, FloatBoost, KLBoost a další. Nutno podoknout, že mnohé varianty jsou stále vytvářený a vylepšovány.



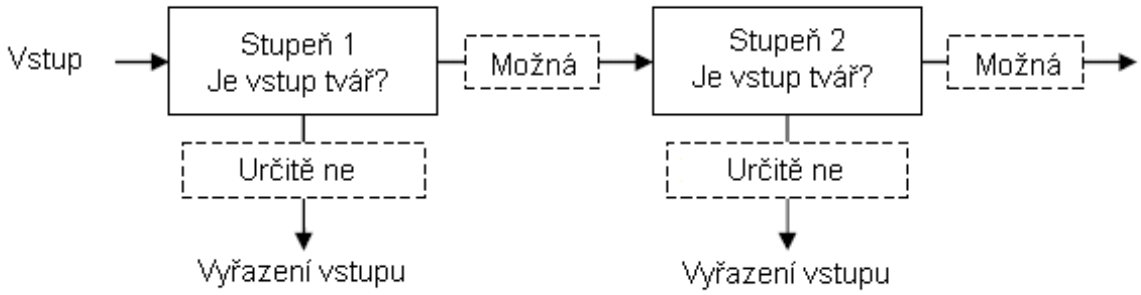
Obrázek 39: Chyba skupiny klasifikátorů

Použití algoritmu AdaBoost může mít obecně několik výhod. Algoritmu pro svůj vstup stačí pouze výcviková množina a sada příznaků. Tím odpadá potřeba mít k dispozici jakoukoliv předešlou znalost o struktuře obličeje. Dalším pozitivem je, že se jedná o adaptivní algoritmus. To znamená, že při výcviku, se algoritmus soustředí na těžce klasifikovatelné vzorky, kterým přiřazuje větší váhu. Na druhou stranu výsledná kvalita klasifikátoru silně závisí na kvalitě tréninkové sady a doba trénování může být zdlouhavá.

7.5 Kaskádová klasifikace

K zjištění tváře v obraze je potřeba prozkoumat všechny možné sub-okna a zjistit jestli obsahují tvář nebo ne. V obyčejném obrázku 320x240 pixelů může být až 500 000 takových sub-oken. Za účelem redukování doby běhu systému se nutné redukovat čas, který systém vynakládá pro každé sub-okno. Za tímto účelem detektor Viola a Jones naznačují použití kaskádových klasifikátorů (obrázek 40). První stupeň kaskády prozkoumá slibné zóny a signalizuje, které sub-okna by měla být ohodnocená příštím stupněm. Jestliže sub-okno je ohodnocené aktuálním klasifikátorem jako *non-face*, pak bude odmítnuté a rozhodování se ukončí. Jinak musí být ohodnoceno dalším klasifikátorem. Pokud sub-okno přežije všechny stupně vodopádu (kaskády), bude označeno jako tvář. Proto se složitost zvyšuje s každým stupněm, ale počet sub-oken, která

mají být ohodnocená, se o mnoho sníží. Struktura vodopádu odráží skutečnost, že uvnitř mnoha jednotlivých snímků převládají v drtivé většině negativní sub-okna. Vodopád se poté pokusí, odmítnou co nejvíce těchto negativních sub-oken pokud možno již v prvních stupních.



Obrázek 40: Kakádové zapojení klasifikátoru

Jednotlivé stupně ve vodopádu jsou konstruovány vytrénovanými klasifikátory za použití algoritmu AdaBoost. Počet stupňů v kaskádě musí, být dostačují k tomu, aby bylo dosaženo dobrých detekčních výkonů při minimalizaci výpočtů. Dané detekční poměry mohou být stanoveny pro každý stupeň kaskády. Pokud je žádoucí mít například detekční poměr (detection rate) kolem 0,9 je možné použít 10 stupňů kaskády, ve které bude mít každý stupeň detekční míru 0,99 ($0,9 \approx 0,99^{10}$). Dosažení takového poměru se může jevit jako obtížný úkol. Nicméně ho odlehčuje skutečnost, že každý stupeň potřebuje docílit poměru chybných přijetí okolo 30% ($0,3^{10} \approx 6x10^{-6}$). Pseudokód na vytvoření kaskády podle Viola - Jones:

1. **Vstup:**
   - f - maximální přijatelné false positive rate v každém stupni kaskády
   - d - minimální přijatelný detection rate v každém stupni kaskády
   - Ftarget - false alarm rate požadovaný na konci procesu
   - N - sada negativních příkladů
   - P - sada pozitivních příkladů
2. **Inicializace:**
   - $F_0 = D_0 = 1$
   - $i = 0$, počet vrstev v kaskádě
3. **Hlavní smyčka:**
   - Dokud není dosažený Ftarget:
   - Přidej novou vrstvu:
   - Dokud pro tuto vrstvu není dosaženo fi, di:
   - Zvyš počet příznaků a trénuj nový silný klasifikátor za pomocí AdaBoost
   - Přepočítej F_i a D_i pro tuto konkrétní vrstvu na testovacích datech

- Přidej nový klasifikátor do kaskády

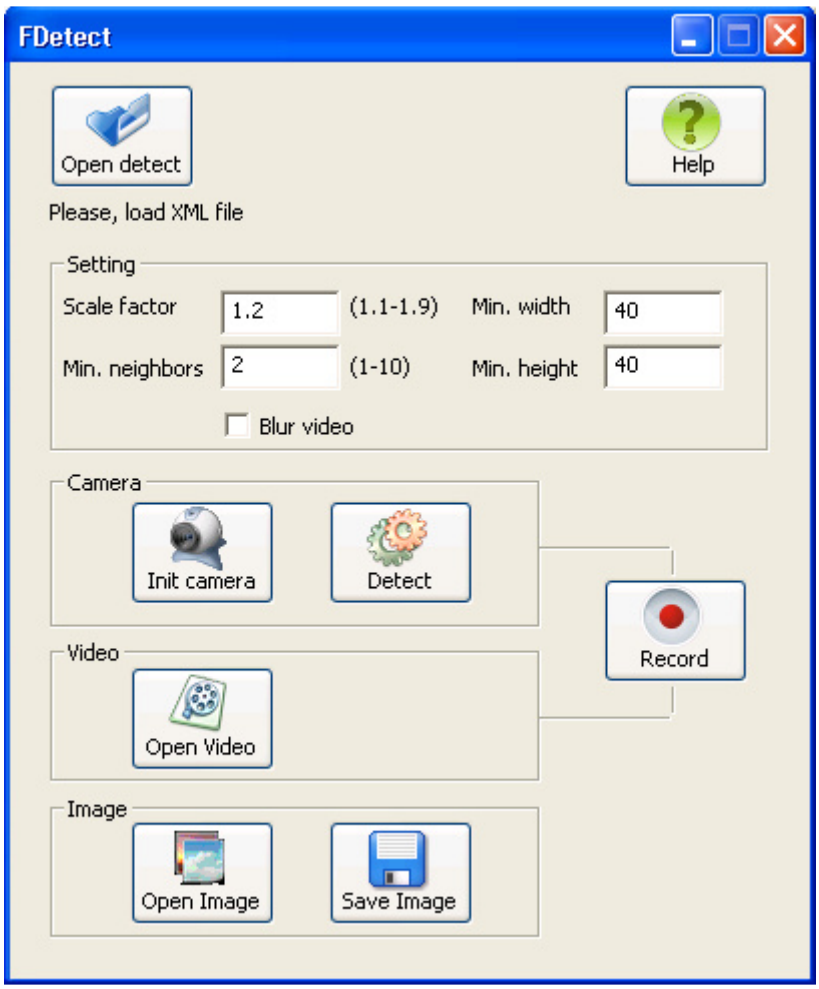
Ve většině případů klasifikátory s více příznaky dosáhnou lepších detekčních objasněností a nižších chybných přijetí. Na druhé straně klasifikátory s více příznaky potřebují více výpočetního času. Jde tedy o to definovat počet stupňů klasifikátoru, počet použitých příznaků v každém stupni a prahové hodnoty pro každý stupeň, tak aby výsledná detekce byla pokud možno co nejefektivnější. O to se budu snažit v experimentální části.

7.6 Skenování detektoru

Finální detektor prochází celý vstupní obraz. Procházení obrazu je umožněno skenujícím okénkem o různých rozměrech. V našem případě bylo zvoleno okno o velikosti 19x19 pixelů a to kvůli parametrům nastavených při samotném trénování klasifikátoru. Jiné standardní velikosti pro detekci tváří mohou být například 24x24 pixelů nebo 20x20 pixelů a další zvolené podle výsledků experimentů. V tomto skenovacím okně jsou poté vypočítány odezvy, které vrátí klasifikátor, na základě kterých je v obraze nalezena tvář. Pro detekování tváří rozličných velikostí je toto skenovací okno zvětšováno až do samotné velikosti obrazu. Parametry detektoru, které ovlivňují celkový výsledek detekce, budou popsány v následující kapitole.

8. Implementace detektoru

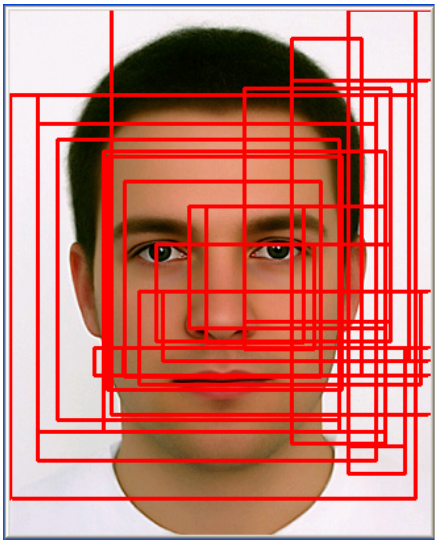
Pro implementaci detektoru Viola a Jones byla využita knihovna OpenCV. Jedná se o volně dostupnou knihovnu napsanou v jazycích C a C++ vyvíjenou firmou Intel. Knihovna může být použita v operačních systémech Windows, Linux, Mac OS X a její vznik se datuje od roku 1999. Knihovna je primárně určena pro práci s obrazy a poskytuje mnoho základních algoritmů pro práci s nimi. OpenCV se stala velmi populární zejména v Číně, Japonsku, Rusku, ale i v Evropě a Izraeli. Knihovna mimo jiné obsahuje algoritmy pro detekci hran, detekci rohů nebo Fourierovu transformaci. Dále knihovna umožňuje práci s kamerou, videem a obsahuje i běžně využívané algoritmy z lineární algebry, statistiky nebo geometrie. OpenCV disponuje vlastním API, pro vytváření a zobrazování jednoduchého GUI. Tato knihovna obsahuje i funkcemi pro podporu právě detektoru Viola-Jones. Za pomocí knihovny tedy byla vytvořena aplikace, která celou detekci obsluhuje (obrázek 41).



Obrázek 41: Uživatelské rozhraní aplikace

Aplikace umožňuje načíst natrénovaný klasifikátor ve formátu XML. Trénování klasifikátoru bylo provedeno také za pomocí nástrojů z OpenCV, které budou vysvětleny později v experimentální části.

Dále aplikace umožňuje nastavení parametrů, které ovlivňují výslednou detekci. Prvním parametrem je tzv. *Scale factor*. Detekce probíhá za pomocí posuvného skenovacího okénka, které se pohybuje napříč obrazem. Aby bylo možné detekovat tváře, které mají různé velikosti je toto okénko zvětšováno až do velikosti vstupního obrazu. Za pomocí parametru *Scale factor,* může být nastaveno o kolik procent, se zvýší velikost skenovacího okénka v každém průchodu. Nastavení na hodnotu 1.2 znamená zvětšení okénka o 20%. Dalším parametrem je tzv. *Minimum neighbors*. Jedná se o parametr, který nastavuje počet sousedních objektů. V okamžiku, kdy detektor lokalizuje tvář, vrátí několik pozitivních odezev na tuto detekci, které se velkou mírou překrývají. V praxi to vypadá jako na obrázku 42.



Obrázek 42: Všechny možné odezvy

Pokud je parametr *Minimum neighbors* nastaven na hodnotu 0, detektor vrátí všechny možné pozitivní detekce, které jsou poté vykresleny za pomocí červeného obdélníku. Z obrázku 34 je patrné, že tyto pozitivní detekce mohou být ve výsledku velmi rušivé a je proto snaha shlukovat je do jednoho výsledného obdélníku. Dostatečně vysoká hodnota parametru zaručí, že skupina pravoúhelníků bude sloučena do menšího počtu nalezením průměrného pravoúhelníku, který by reprezentoval tuto skupinu. Pokud by se ovšem parametr nastavil na velmi vysokou hodnotu, hrozilo by možné ignorování opravdových odezev na tvář. V aplikaci není hodnota 0 tohoto parametru povolena.

Další v pořadí jsou parametry *Minimum width a Minimum height.* Jedná se o nastavení minimální velikosti skenovacího okénka. Tváře menší jak tato velikost poté nebudou detekovány. Aplikace umožňuje detekovat obraz z kamery, obrázků i videa.

Samotná aplikace byla vytvořena v jazyce C++ a jedná se o Win32 aplikaci, která se skládá z jednoho hlavního okna. V tomto okně jsou odchytávány všechny události potřebné pro běh samotné aplikace. Obecně těmito událostmi může být například stisk klávesnice nebo akce za pomocí myši. Vytvořenému oknu aplikace je poté zaslána systémem zpráva, která danou událost popisuje a okno na ně může nebo nemusí určitým způsobem reagovat. Jedná se tedy o událostmi řízené programování. V další kapitole bude popsáno samotné vytvoření klasifikátoru.

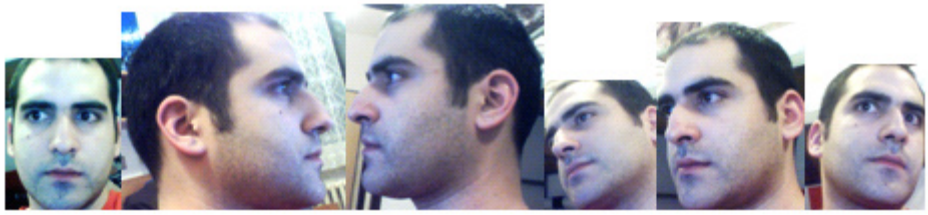
9. Experimenty

Pro proces učení je nutné mít k dispozici sadu pozitivních a negativních snímků. Sada pozitivních snímků reprezentuje požadovaný objekt pro detekci, naopak v sadě negatiních snímků by tento objekt neměl být zastoupen. OpenCV disponuje nástroji pro natrénování klasifikátoru právě z již zmíněných pozitivních a negativních snímků. Podrobný popis a práce s nástroji je popsána v příloze A.

9.1 Vlastní trénování

První pokusy o natrénování klasifikátoru byly provedeny na fotografiích mé tváře. Zvolil jsem následující metodiku. Vyfotil jsem svoji tvář z šesti různých úhlů a za použití utility *createsamples* jsem každou fotografii rozmnožil v prvním pokusu na 200 snímků. Série fotografií je na obrázku 43. Nastavení jsem zvolil následující:

createsamples -img 01.png -num 200 -bg negative.txt -vec samples01.vec -maxxangle 0.6 -maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 255 -bgthresh 0 -w 30 -h 30



Obrázek 43: Trénovací množina – pozitivní příklady

V tomto okamžiku jsem získal 1200 pozitivních snímků. Dalším krokem bylo natrénování klasifikátoru. Tady jsem při prvním pokusu zvolil počet negativních snímků 2000, stupňů kaskády 15, algoritmus pro trénování jsem nenastavoval a ponechal standardní nastavení, které zvolí Gentle AdaBoost. Celý výpis hodnot:

haartraining -data haarcascade -vec samples.vec -bg negative.txt -nstages 15 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 1200 -nneg 2000 -w 30 -h 30 -nonsym -mem 1024 -mode ALL

Před samotným testováním výkonnosti klasifikátoru jsem za pomocí utility *createsamples* nechal vygenerovat na různých pozicích s různými světelnými změnami obrázek mé tváře, který

byl náhodně vložen do testovacích snímků. Tak byl vytvořen soubor popisující souřadnice mé tváře v obrazech.

Po natrénování klasifikátoru jsem přešel k jeho testování. Pro testování byla vytvořena čtveřice různých konfigurací. Konfigurace jsou uvedeny v tabulce 1. Dále byla vytvořena sada 50 testovacích snímků. Jednotlivé úspěšnosti detekcí v daných konfiguracích jsou uvedeny v tabulce 2.

	Scale factor	*Min. neighbors*	*Size (width, height)*
Konfigurace 1	1.1	3	30, 30
Konfigurace 2	1.2	2	30, 30
Konfigurace 3	1.3	1	30, 30
Konfigurace 4	1.1	4	30, 30

Tabulka 1: Nastavení konfigurací pro testování.

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	47	6	88%
Konfigurace 2	48	21	69%
Konfigurace 3	25	29	46%
Konfigurace 4	44	9	83%

Tabulka 2: Úspěšnost detekcí u jednotlivých konfigurací.

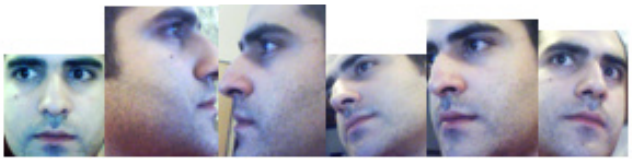
Celková úspěšnost v tabulce 2 je počítána jako relativní poměr správných rozhodnutí detektoru podle vzorce (17):

$$A_{cc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

V tabulce je poté *Správné rozhodnutí* = TP +TN a *Chybné rozhodnutí* = FP + FN. Kde TP (True Positive, správné zařazení), TN (True Negative, správné odmítnutí), FP (chybné zařazení), FN (False Negative, chybné odmítnutí).

Z výsledků, které jsem získal, bylo vidět, že úspěšnost konfigurace 1 se jeví jako nejlepší nastavení parametrů detektoru. Nicméně v dalším kroku se pokusím navýšit počet příkladů pro učení za účelem zlepšení celkové úspěšnosti.

Šestici pozitivních snímků z minulého trénování jsem pozměnil (obrázek 44) a počet vygenerovaných pozitivních příkladů jsem zvýšil na 300. Tímto krokem jsem měl nyní k dispozici 1800 pozitivních snímků. Zároveň jsem zvýšil počet negativních příkladů na 2800. Kromě těchto změn bylo vše ponecháno jako v předchozím případě. Vygenerovaný klasifikátor byl poté opět použit na detekci. Výsledek stejně nastavených konfigurací jako v předešlém případě a jejich úspěšnosti jsou zobrazeny v tabulce 3.



Obrázek 44: Druhá trénovací množina – pozitivní příklady

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	50	10	83%
Konfigurace 2	49	27	64%
Konfigurace 3	42	18	70%
Konfigurace 4	50	4	92%

Tabulka 3: Úspěšnost detekcí u jednotlivých konfigurací.

Z výsledků, které jsem dostal, můžu usuzovat, že navýšení trénovacích dat mělo pozitivní dopad na celkovou úspěšnost detekce. Při konfiguraci 4 se úspěšnost detekce dostala až nad hranici 90%.

Tato metodika by se dala použít, pokud by byla potřeba detekovat jen určitý jeden objekt nebo sadu objektů. Za pomocí této metodiky by se poté daly detekovat tváře lidí jen v určitém zařízení. Díky tomu by byla možnost sledovat tváře jen určitých lidí, například lidí z databáze zločinců. V okamžiku kdyby se obličej takového člověka ukázal na kameře, byl by detekován a mohla by být zaslána výstražná zpráva příslušné osobě. Ovšem značná nevýhoda této detekce vznikne v případě, kdy daná osoba významně změní svůj vzhled. V takovém případě se tato metoda jeví jako nepoužitelná.

Dále jsem testoval trénování obecně, aby byla možnost detekovat všechny obličeje. K tomu aby se daly detekovat pokud možno všechny tváře je nutné klasifikátor naučit na velké množství různých obličejů. To mě postavilo před problém, kde takové tváře sehnat? Podařilo se mi nalézt hned několik institucí, které poskytují databáze tváří pro vědecké účely zcela zdarma. První databázi, na které jsem otestoval trénování, byla MIT CBCL Face Database[27].

9.2 MIT CBCL Face database

Jedná se o databázi, která obsahuje 2429 pozitivních snímků, tedy snímků tváří a 4548 negativních snímků, tedy snímků, kde se tvář nevyskytuje. Všechny snímky měly velikost 19x19 pixelů. Na této velikosti se poté zakládalo i vlastní trénování. Ukázka tváří z databáze je na obrázku 45.

Obrázek 45: Ukázka obličejů z MIT CBCL Face Database

Nastavení jsem zvolil následující:

haartraining -data haarcascade -vec samples.vec -bg negative.txt -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 2400 -nneg 2400 -w 19 -h 19 -nonsym -mem 1024 -mode ALL > train.txt

Počet stupňů kaskády byl nastaven na 20, počet vybraných příznaků v každé iteraci na 2, počet pozitivních příkladů na 2400 a počet negativních příkladů na 2400. Protože se tváře v databázi vyskytovaly s velikostí 19x19 pixelů, tak i velikost při trénování byla zvolena na 19x19. Pro testování jsem si zvolil následující referenční obrázek (obrázek 46), který obsahuje 40 obličejů lidí různých národností a pohlaví. Snaha tedy byla o co možná nejvíce detekovaných tváří v tomto snímku.

Obrázek 46: Testovací snímek

Nastavení konfigurací jsem ponechal stejné jako v předchozích případech, jen velikost jsem změnil na 19x19. Celková úspěšnost bude počítána jako poměr detekovaných tváří ke všem tvářím v obrázku. Z toho vyplívá, že *správné rozhodnutí* v tabulce 4 odpovídá počtu správně detekovaných tváří a *chybné rozhodnutí* značí počet tváří, které detekovány nebyly.

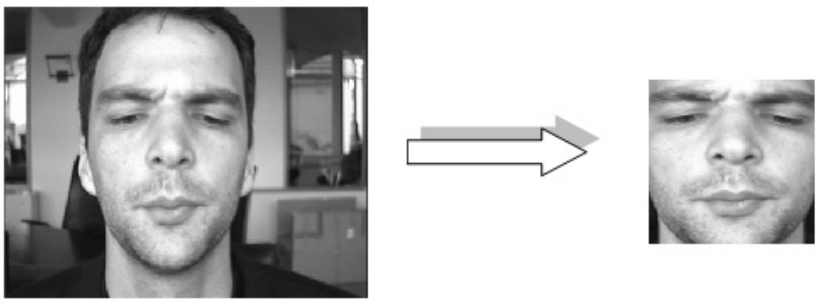
	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	20	20	50%
Konfigurace 2	16	24	40%
Konfigurace 3	0	40	0%
Konfigurace 4	16	24	40%

Tabulka 4: Úspěšnost detekcí u jednotlivých konfigurací - CBCL

V další fázi jsem se pokusil zvýšit počet negativních snímků o 1000. Tedy celkový počet negativních snímků pří trénování byl 3400. Bohužel žádný významný efekt to u referenčního snímku nemělo. Navýšením počtu negativních snímků nebylo dosaženo požadovaného efektu zvýšit počet detekovaných tváří, proto jsem se rozhodnul zvýšit počet pozitivních snímků. Bylo tedy nutné nalézt další databázi tváří. Těmi se staly databáze BioID face database[28] a Caltech frontal face dataset[29].

9.3 CBCL + BioID + Caltech

BioID je další volně dostupná databáze tváří. Její autoři uvádějí, že při sestavování dbaly na různorodost osvětlení a pozadí. Databáze obsahuje zhruba 1500 snímků obličejů, které jsem pro samotné trénování musel upravit. Ukázka snímku a jeho následná úprava je na obrázku 47. Jedná se o úpravu pozadí, kde z každého obrázku bylo třeba extrahovat jen obličej.



Obrázek 47: Úprava snímků. Na levé straně originální snímek z BioID databáze. Na straně pravé snímek po úpravě, který byl použit při trénování.

Caltech databáze na tom s úpravou byla podobně a celkově databáze osahovala 450 tváří. Nyní bylo k dispozici pro trénování cca. 4400 snímků tváří. Počet negativních snímků byl také značně navýšen. V této chvíli mohlo být spuštěno další trénování. Výsledný skript tedy vypadal následovně:

haartraining -data haarcascade -vec samples.vec -bg negative.txt -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 4400 -nneg 8040 -w 19 -h 19 -nonsym -mem 1024 -mode ALL

Nastavení bylo ponecháno obdobné jako v předchozím případě až na počet příkladů. Při testování byly použity opět stejné konfigurace a stejný referenční snímek z obrázku 46. Výsledky jsou zobrazeny v tabulce 6. Do tabulky byla přidána konfigurace 5 (tabulka 5), která se pro tento klasifikátor jevila jako optimální. Konfigurace byla získána dalším experimentováním s parametry detektoru.

	Scale factor	*Min. neighbors*	*Size (width, height)*
Konfigurace 5	1.1	1	19, 19

Tabulka 5: Dodatečná konfigurace

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	29	11	72%
Konfigurace 2	25	15	62%
Konfigurace 3	19	21	47%
Konfigurace 4	25	15	62%
Konfigurace 5	33	7	82%

Tabulka 6: Úspěšnost detekcí u jednotlivých konfigurací CBCL + BioId + Caltech

Jak již předešlý odstavec nastínil, konfigurace 5 se pro tento klasifikátor jeví jako optimální. S takovou konfigurací bylo možné dosáhnout detekce až 33 tváří z požadovaných 40, tedy procentuální úspěšnost se pohybuje na hranici 82%. Na následujícím snímku 48 je výsledek detekce při konfiguraci 5. Pro pozdější testování bude označen tento klasifikátor jako číslo 1.

Obrázek 48: Výsledek detekce

Jak si povede detektor s obrázkem, kde nejsou pouze tváře, ale značné zastoupení může mít i pozadí nebo oblečení osob? Další test byl proveden na následujícím snímku 49. Tento test byl proveden ke zjištění, jestli nebudou při detekci převažovat chybné označení.

Obrázek 49: Testovací snímek

Jinými slovy, jestli například detektor neoznačí za tvář kousek zdi nebo jiné předměty, které jako tváře být označeny nemají. Konfigurace byly ponechány stejné jako v předchozím případě.

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	15	4	78%
Konfigurace 2	13	6	68%
Konfigurace 3	5	14	26%
Konfigurace 4	14	5	73%
Konfigurace 5	18	1	89%

Tabulka 7: Úspěšnost detekcí u jednotlivých konfigurací CBCL + BioId + Caltech, druhý testovací snímek

Z tabulky 7 je zřejmé, že nejlépe si opět vedla konfigurace 5. Výsledný obrázek detekce je vidět na obrázku 50. Nutno podotknout, že dvojitou detekci jsem v tomto případě nepovažoval za chybnou. Důležité v tomto kroku testování bylo, že detektor neoznačil ani v jedné konfiguraci jako tvář pozadí. Z toho lze usuzovat, že počet a výběr negativních snímků se jeví jako dostatečný. Nicméně jistě se najdou snímky, které by toto tvrzení vyvrátily. Je zřejmé, že bude záležet na tom, v jakém prostředí bude detektor použit.

Obrázek 50: Druhý testovací snímek, konfigurace 5

Další experimenty byly provedeny s navýšením použitých příznaků v klasifikátorech. Doposud byl slabý klasifikátor složen ze dvou příznaků. Tento počet byl v dalším kroku navýšen na příznaky čtyři. Ostatní nastavení bylo ponecháno stejné jako v předchozím případě. Výsledný skript tedy vypadal následovně:

haartraining -data haarcascade -vec samples.vec -bg negative.txt -nstages 20 -nsplits 4 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 4400 -nneg 8040 -w 19 -h 19 -nonsym -mem 1024 -mode ALL

V tabulce 8 jsou uvedeny výsledky detekce za pomocí takového klasifikátoru.

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	16	3	84%
Konfigurace 2	12	7	63%
Konfigurace 3	1	18	26%
Konfigurace 4	14	5	5%
Konfigurace 5	17	5	72%

Tabulka 8: Úspěšnost detekcí u jednotlivých konfigurací CBCL + BioId + Caltech, druhý testovací snímek, navýšení příznaků

Z tabulky 8 je vidět, že navýšení počtu použitých příznaků při konstrukci slabých klasifikátorů nevedlo k tíženému výsledku. Dokonce se v detekci při konfiguraci 5 objevily tři chybné detekce (obrázek 51), kdy byly jako tváře označeny části oblečení. Pro pozdější testování bude označen tento klasifikátor jako číslo 2.

Obrázek 51: Druhý testovací snímek, navýšení příznaků, konfigurace 5

Navíc toto nastavení nepřineslo zlepšení, ani v prvním testovacím snímku. Pokud tedy nepomohlo zlepšení navýšením použitých příznaku ve slabých klasifikátorech, mohlo by naopak pomoci jejich zjednodušení na příznak jeden. Skript se lišil opět pouze v počtu použitých příznaků v klasifikátorech:

haartraining -data haarcascade -vec samples.vec -bg negative.txt -nstages 20 -nsplits 1 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 4400 -nneg 8040 -w 19 -h 19 -nonsym -mem 1024 -mode ALL

Výsledek při použití na druhém testovacím snímku je uveden v následující tabulce 9. Pro pozdější testování bude označen tento klasifikátor jako číslo 3.

	Správné rozhodnutí	*Chybné rozhodnutí*	*Celková úspěšnost*
Konfigurace 1	17	2	89%
Konfigurace 2	15	4	78%
Konfigurace 3	6	13	31%
Konfigurace 4	16	3	84%
Konfigurace 5	18	1	94%

Tabulka 9: Úspěšnost detekcí u jednotlivých konfigurací CBCL + BioId + Caltech, druhý testovací snímek, snížení příznaků

Při porovnání tabulek 7, 8 a 9 je zřejmé, že při všech konfiguracích došlo k jednoznačnému zlepšení celkové úspěšnosti právě při snížení počtu příznaků použitých ve slabých klasifikátorech. Dokonce po dalším experimentování s parametry detektoru se podařilo najít pro tento postup trénování optimální konfiguraci uvedenou v tabulce 10.

Obrázek 52: Druhý testovací snímek, nová konfigurace, snížení příznaků

	Scale factor	*Min. neighbors*	*Size (width, height)*
Konfigurace 6	1.1	2	19, 19

Tabulka 10: Druhá dodatečná konfiguace

Při této konfiguraci se podařilo na druhém testovacím snímku detekovat všechny tváře bez jediné chybné detekce (obrázek 52). Nicméně při testováním na prvním snímků k žádnému navýšení detekovaných tváří nedošlo ani při dodatečné konfiguraci. Dokonce se objevily na prvním testovacím snímku nechtěné falešné detekce, proto bych snížení počtu příznaků nepovažoval za ideální řešení pro celkové zlepšení detekce.

Kompletní trénování samozřejmě trvá určitý čas. Vzhledem k tomu, že trénování bylo vykonáváno na různých stanicích, neuvádím zde přehled a porovnání celkových časů, při výcviku. Nutno ale podotknout, že na stroji s procesorem Intel Core 2 Duo E7500 s 2GB RAM trval proces trénování okolo 48 hodin.

Z testování je vidět, že opět záleží na tom, v jakém prostředí by měl být detektor použit. Při stejném nastavení může detektor pracovat jinak na různých snímcích v různě odlišných pozadích. Důležitým faktorem je ve všech případech počet použitých výcvikových dat. Použití samotné CBCL databáze, která čítala na 2400 snímků tváří, se jevila pro praktické použití, při tomto druhu trénování jako nepoužitelná. Odlišným případem bylo zkombinování dalších databází tváří. Při tomto postupu bylo detekováno mnohem více obličejů a celkový výsledek byl o poznání lepší. Experimentování s různým počtem příznaků ve druhém testovacím snímku sice vedlo k dílčím zlepšením, ale na prvním testovacím snímku selhávalo. Tedy počet detekovaných tváří nebyl navýšen. Z výše uvedených testů a závěrů lze usuzovat, že největší podíl na úspěšnosti detekce mají právě výcviková data a to především jejich správná úprava a výběr. V příloze B jsou zobrazeny výsledky detekce jednotlivých klasifikátorů na vybraných snímcích z testovací databáze CMU, při své nejlepší konfiguraci podle druhého testovacího snímku.

10. Závěr

Dle zadání byly v teoretické části popsány vybrané metodiky pro detekci tváří v obrazech. Z těchto metodik byl vybrán algoritmus Viola and Jones, který byl použit pro vytvoření aplikace. Tento algoritmus má pro svou praktickou realizaci velkou podporu v knihovně OpenCV. Její možné využití bylo také součástí zadání. Ta navíc disponuje nástroji pro natrénování klasifikátoru podle přístupu Viola and Jones. Takový klasifikátor se poté dá použít pro samotnou detekci.

Po vytvoření aplikace, která umožňuje načtení klasifikátoru natrénovaného za pomocí nástrojů z OpenCV mohlo být provedeno testování. Ještě před samotným testováním byly představeny různé postupy, jakým způsobem takový klasifikátor vytvořit. Po jeho vytvoření již mohly být spuštěny série testů, které zjišťovaly úspěšnosti jednotlivých klasifikátorů. Testování bylo provedeno jak vzhledem k nastavení samotné vytvořené aplikace, tak vzhledem k metodice trénování, která byla zvolena.

Z testů se poté dalo usoudit, že nejvýznamnější roli na samotnou úspěšnost klasifikátoru měla výcviková data. V práci byly představeny volně dostupné databáze tváří a jejich možné využití při práci s nástroji, kterými disponuje OpenCV. V příloze A jsou tyto nástroje představeny i s praktickými ukázkami. Kromě výcvikových dat úspěšnost detekce ovlivňuje i nastavení parametrů samotné aplikace. Pro otestování vlivu těchto parametrů, bylo vytvořeno několik konfigurací, které byly postupně otestovány a jejich výsledky porovnány.

Popsaná a zvolená metodika může být dále vylepšována navýšením výcvikových dat a dalším experimentováním s nastaveními, které ovlivňují samotný proces trénování. Aplikace i přesto, že byla vyvíjena pro účel testování, může být použita i v praktických úlohách. V okamžiku, kdy jsou k dispozici tváře v obraze, může být provedena například jejich anonimizace, uložení nebo může být počítán jejich počet v daných obrazech, pro další vyhodnocení.

Knihovna OpenCV obsahuje již natrénované klasifikátory. Jedná se o klasifikátory naučené na rozpoznávání také tváří nebo lidských postav. Celý postup trénování klasifikátoru, který je popsán v experimentální části může být použit i pro jiné účely než jen pro samotnou detekci tváří nebo postav a uživatel tedy nemusí být na již přiložených klasifikátorech z OpenCV závislý. Za předpokladu, že by byla k dispozici potřebná výcviková sada pozitivních i negativních snímků lze postup aplikovat víceméně na jakýkoliv zájmový objekt. Shromážděná databáze tváří, která byla použita pro tuto práci, může být dále využita při dalších experimentech.

Literatura:

[1] YANG, Ming-Hsuan. *Recent Advances in Face Detection. California*, USA : Honda Research Institute, 2004. Dostupné z WWW: <http://vision.ai.uiuc.edu/mhyang/papers/icpr04_tutorial.pdf>

[2] YOW, Kin Choong; CIPOLLA, Roberto. *Feature-Based Human Face Detection* [online]. University of Cambridge, August 1996 [cit. 2010-05-05]. Dostupné z: <http://mi.eng.cam.ac.uk/~cipolla/publications/article/1997-IVC-face-detection.pdf>

[3] AHMAD, *Naveed. Association for Computing Machinery* [online]. 2003 [cit. 2010-03-14]. The Humanoid Robot Cog. Dostupné z: <http://www.acm.org/crossroads/xrds10-2/robotcog.html>

[4] SINHA, Pawan. *Qualitative Representations for Recognition* [online]. Cambridge : Massachusetts Institute of Technology, 2002 [cit. 2010-03-14]. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.5960&rep=rep1&type=pdf>

[5] DRÁBEK, O.; SEIDL, P. ; TAUFER, I. UMĚLÉ NEURONOVÉ SÍTĚ : ZÁKLADY TEORIE A APLIKACE (3). *CHEMagazín • Číslo 1 • Ročník XVI.*, 2006. [cit. 20010-01-10]. Dostupné z: <www.chemagazin.cz/Texty/CHXVI_1_cl3.pdf>

[6] HORKÝ, L.; BŘINDA, K. *Neuronové sítě*, 2008-2009 [cit. 2010-03-21]. Dostupné z: <http://fyzsem.fjfi.cvut.cz/2008-2009/Leto09/proc/neurony.pdf>

[7] ROWLEY, Henry A.; BALUJA, Shumeet; KANADE, Takeo. *Neural Network-Based Face Detection* [online]. January 1998 [cit. 2010-03-21]. Dostupné z: <http://www.informedia.cs.cmu.edu/documents/rowley-ieee.pdf>

[8] KRUEGER, Jon; ROBINSON, Marshall B. *Connexions* [online]. 2004 [cit. 2010-03-26]. Obtaining the Eigenface Basis. Dostupné z: <http://cnx.org/content/m12531/latest/>

[9] DELAC, K.; GRGIC, M.; GRGIC, S. Statistics in Face Recognition: Analyzing Probability Distributions of PCA, ICA and LDA Performance Results, *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, 2005, Zagreb, Croatia, 15-17 September 2005. Dostupné z: <http://www.face-rec.org/algorithms/Comparisons/delac_01.pdf>

[10] LONG, Quoc. *CS Study Fun* [online]. 2008 [cit. 2010-03-26]. Linear discriminant analysis. Dostupné z: <http://csstudyfun.etintin.com/tag/maximum-likelihood-estimation/>

[11] ŽIŽKA, Jan. Support vector machines (SVM) [online], 2005. 4 s. Studijní materiály předmětu FI:PA034. Masarykova Univerzita. Dostupné z: <http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf?fakulta=1433;obdobi=3523;kod=PA034>

[12] OSUNA, Edgar; FREUND, Robert; GIROSI, *Federico Training Support Vector Machines: an Application to Face Detection*. 1997 [cit. 2010-03-26]. Dostupné z: <http://cbcl.mit.edu/projects/cbcl/publications/ps/cvpr97-face.ps.gz>

[13] TAKAHASHI, Norikazu. *Norikazu Takahashi Personal Web Page* [online]. Efficient Learning Algorithms for Support Vector Machines 2004 [cit. 2010-03-26]. Dostupné z: <http://www-kairo.csce.kyushu-u.ac.jp/~norikazu/research.en.html>

[14] HEARST, Marti A. Trends & Controversies: Support Vector Machines. *IEEE Intelligent Systems*. 1998 [cit. 2010-03-26]. Dostupné z: <http://www.cs.toronto.edu/~zemel/Courses/CS411/Papers/svmOverview.pdf>

[15] PAVELKA, T.; EKŠTEJN, K.; ANDRŠ, D. *Hybridní rozpoznávač přirozené řeči pro český jazyk, proc. Kognícia a umelý život,* 2005, Smolenice, Slovakia, 2005 Dostupné z: <http://hilbert.chtf.stuba.sk/KUZV/download/kuzv-pavelka-ekstein-andrs.pdf>

[16] KADOURY, S. *Face Detection using Locally Linear Embeddings*, Master Thesis, McGill Univ., 2005 Dostupné z: <http://www.cim.mcgill.ca/~levine/thesis_SamuelKadoury.pdf>

[17] YANG,M. H.,KRIEGMAN, D., AND AHUJA,N. 2002. Detecting faces in images: A survey. *IEEE Trans.* Dostupné z: <http://www.computer.org/portal/web/csdl/doi/10.1109/34.982883>

[18] POLIKAR, Robi, *Pattern recognition*, Rowan University, Glassboro, New Jersey 2007, Dostupné z: <http://users.rowan.edu/~polikar/RESEARCH/PUBLICATIONS/wiley06.pdf>

[19] http://www.arr.cz/userfiles/image/Uspesna_zena/men-vs-women.jpg

[20] AKSOY, Selim, *Introduction to Pattern Recognition*, Spring 2010 [cit. 2010-03-27]. Dostupné z: <http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551/slides/cs551_intro.pdf>

[21] http://img.brothersoft.com/screenshots/softimage/v/verifinger_standard_sdk_trial-52747-1.jpeg

[22] JORGENSEN, A. *AdaBoost and Histograms for Fast Face Detection*, 2006. Dostupné z: <http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2006/rapporter06/jorgensen_anders_06110.pdf>

[23] PAVLOVIC, V. Efficient Detection of Objects and Attributes using Boosting. *IEEE Conf. Computer Vision and Pattern Recognition*. Dostupné z: <http://www.cs.rutgers.edu/~vladimir/pub/cvpr01.pdf>

[24] WANG, Chieh-Chih. VASC Image Database. 1997. *CMU Image Database*. Dostupné z: <http://vasc.ri.cmu.edu/idb/html/face/index.html>

[25] FREUND, Yoav; E. SCHAPIRE, Robert. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. *Journal of computer and system sciences* [online]. 1997, 55, [cit. 2010-05-05]. Dostupný z: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=50856377919EFF3A082577189BCCC305?doi=10.1.1.109.5335&rep=rep1&type=pdf>

[26] PALLA, Kostantina. *Robust Real-time Face Detection* [online]. 2009. Prezentation. University of Edinburgh. Dostupné z: <http://www1.cs.columbia.edu/~belhumeur/courses/biometrics/2009/violajones.ppt>

[27] *Center for Biological & Computational Learning (CBCL)* [online]. 2000 [cit. 2010-04-04]. Face Data. Dostupné z: <http://cbcl.mit.edu/projects/cbcl/software-datasets/FaceData1Readme.html>

[28] *BioID face database* [online]. 2010 [cit. 2010-04-04]. BioID. Dostupné z: <http://www.bioid.com/support/downloads/software/bioid-face-database.html>

[29] *California Institute of Technology* [online]. 2010 [cit. 2010-04-04]. Caltech categories. Dostupné z: <http://www.vision.caltech.edu/html-files/archive.html>

Přílohy

A) Práce s nástroji pro trénování

Pozitivní sada snímků

V prvním kroku je tedy třeba vytvořit kolekci pozitivních snímků. Tady se nabízí dvě možnosti jak takovou množinu vytvořit. První možností je použít přístup, ve kterém každý pozitivní snímek musí mít následující formát:

Název_souboru_1 počet x11 y11 w11 h11 x12…
Název_souboru_2 počet x21 y21 w21 h21 x22…

Kde každý řádek musí začínat názvem souboru (obrázku), následovaného počtem sledovaných (pozitivních) objektů v něm. Dále je nutné, aby soubor obsahoval souřadnice levého horního bodu, šířku a výšku pozitivního objektu. Ukázka takového souboru (deskriptoru):

/01Train/positive/01p.bmp 1 282 169 183 121
/01Train/positive/02p.bmp 1 23 11 182 121

Za předpokladu, že je k dispozici takový deskriptor popisující pozitivní snímky pro trénování, lze spustit utilitu *createsamples*, která z něj vytvoří .vec soubor:

createsamples -info samples.dat -vec samples.vec -w 24 -h 24

Parametry:
-info <deskriptorový soubor popisující pozitivní snímky>
-vec <název výstupního .vec souboru>
-w -h <pozitivní vzorky extrahované z obrazů budou nastaveny na stejnou velikost, v tomto případě na velikost 24x24>

Druhý přístup využívá pouze jeden pozitivní snímek, který rozmnoží za pomocí zkroucení a změny světlosti z něj vytvoří sadu pozitivních výcvikových vzorků, opět za použití nástroje *createsamples*. Příklad na vytvoření takové sady vypadat následovně:

createsamples -img face01.jpg -num 100 -bg negatives.dat -vec samples.vec -maxxangle 0.6 -maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 0 -bgthresh 0 -w 24 -h 24

Parametry:
-img <jeden pozitivní vzorek>
-num <počet výsledných vzorků>
-bg <kolekce snímků pozadí>
-vec <název výstupního .vec souboru>
-maxxangle <nejvyšší míra rotace na ose x>
-maxyangle < nejvyšší míra rotace na ose y>
-maxzangle < nejvyšší míra rotace na ose z>
-maxidev <odchylka změny světlosti>
*-**bgcolor** -**bgthresh*** <značí transparentní barvu, všechny pixely mezi hodnotami *bgcolor a bgthresh* budou označeny za transparentní>

Obě metody na vytvoření pozitivní sady snímků jdou různě kombinovat, stejně tak lze i spojovat vytvořené .vec soubory do jednoho. To může mít užitek v okamžiku, kdy je k dispozici nasnímaný detekovaný objekt na více snímcích. V tom případě lze použit druhou metodu a vytvořit pro každý snímek jeden .vec soubor.

Negativní sada snímků

Dále je nutné mít k dispozici soubor popisující tzv. negativní sadu obrazů. Lze použít libovolné obrazy, které neobsahují detekovaný objekt. Soubor musí obsahovat cesty k těmto negativním vzorkům. Může mít například následující formát:

negative/n01.jpg
negative/n02.jpg
negative/n02.jpg

Trénování

Pokud jsou k dispozici trénovací data, je možné za použití *haartraining* utility trénovat vlastní klasifikátor. Skript pro trénování může vypadat následovně:

haartraining -data haarcascade -vec samples.vec -bg negatives.dat -nstages 20 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 1000 -nneg 2000 -w 20 -h 20 -nonsym -mem 1024

Parametry:

-data <název výstupního adresáře, ve kterém budou uloženy data klasifikátoru>

-vec <název vec souboru, ve kterém jsou popsány pozitivní příklady>

- bg <sada negativních snímků>

-nstages <počet trénovacích stupňů>

-minhitrate <minimální hodnota úspěšnosti detekce jednotlivých úrovní kaskády>

-falsealarm <maximální hodnota nesprávných detekcí v jednotlivých úrovní kaskády>

-mode <složí k výběru haarových příznaků>

-w -h <velikost výcvikových vzorků, musí mít stejnou velikost jako při vytváření utilitou *trainingsamples*>

-npos -nneg <počet pozitivních a negativních vzorků použitých při trénování v každém stupni klasifikátoru>

-nonsym <pokud nejsou pozitivní příklady vertikálně symetrické, v opačném případě se nastaví ***-sym***>

-mem <dostupná paměť pro výpočty v MB>

K dispozici je ještě další sada parametrů, které mohou být použity pro zlepšení výsledné klasifikace. Jsou to například parametry:

-bt <slouží k nastavení trénovacího algoritmu, k dispozici jsou algoritmy: Discrete Ada Boost, Real AdaBoost, LogitBoost, Gentle AdaBoost>

-nsplits <počet příznaků použitých ve slabých klasifikátorech>

Ukázka výstupu:

```
Data dir name: haarcascade
Vec file name: samplescollection.vec
BG  file name: negative.txt
Num pos: 1200
Num neg: 1500
Num stages: 15
Num splits: 2 (tree as weak classifier)
Mem: 1024 MB
Symmetric: FALSE
Min hit rate: 0.999000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 30
Height: 30
Max num of precalculated features: 66280
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 3.05176e-005

Tree Classifier
Stage
```

```
+---+
|  0|
+---+

Number of features used : 642592

Parent node: NULL

*** 1 cluster ***
POS: 1200 1200 1.000000
NEG: 1500 1
BACKGROUND PROCESSING TIME: 0.02
Precalculation time: 17.86
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.933397| 1.000000| 1.000000| 0.062593|
+----+----+-+---------+---------+---------+---------+
|   2|100%|-|-1.892786| 1.000000| 1.000000| 0.041111|
+----+----+-+---------+---------+---------+---------+
|   3|100%|-|-1.924738| 0.999167| 0.164000| 0.020741|
+----+----+-+---------+---------+---------+---------+
Stage training time: 1102.77
Number of used features: 6
```

Testovací sada snímků

Po natrénování klasifikátoru je žádoucí otestovat kvalitu jeho klasifikace. Pokud není k dispozici testovací množina snímků, je možné opět za použití utility *createsamples*, vytvořit sadu testovacích obrazů z jednoho pozitivního příkladu, který je vložen na různé pozice do sady snímků, které se nazývají snímky pozadí. Skript pro vygenerování testovacích snímků:

createsamples -img face01.jpg -num 100 -bg negatives.dat -info test.dat -maxxangle 0.6 -maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 0 -bgthresh 0

Parametry:

-img <jeden pozitivní vzorek>

-num <počet výsledných vzorků>

-bg <kolekce snímků pozadí>

-info <název výstupního deskriptorového souboru>

-maxxangle <nejvyšší míra rotace na ose x>

-maxyangle < nejvyšší míra rotace na ose y>

-maxzangle < nejvyšší míra rotace na ose z>

-maxidev <odchylka změny světlosti>

*-**bgcolor*** ***-bgthresh*** <značí transparentní barvu, všechny pixely mezi hodnotami *bgcolor a bgthresh* budou označeny za transparentní>

Testování výkonu

Posledním krokem je otestování výkonnosti výsledného klasifikátoru utilitou *performance*. Z předchozího kroku jsou k dispozici testovací snímky, které jsou popsány deskriptorovým souborem. Ten je při testování výkonu použit jako vstup. Jako výstup poté dostaneme počet správně nalezených objektů, množství zmeškaných objektů a počet falešných signalizací. Skript pro testování výkonu:

performance -data haarcascade -w 30 -h 30 -info tests.dat

Parametry:

- *data* <název adresáře, ve kterém je natrénovaný klasifikátor uložen>

- *info* <název souboru, který popisuje testovací snímky>

-w -h < velikost výcvikových vzorků, musí mít stejnou velikost jako při vytváření utilitou *trainingsamples* >

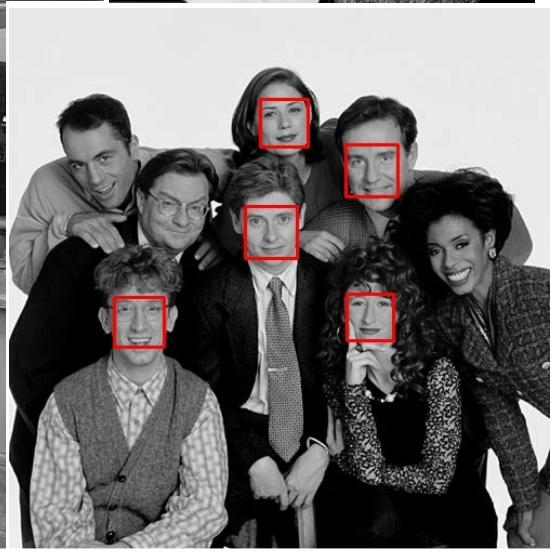
Ukázka výstupu:

```
+================================+======+======+======+
|            File Name           | Hits |Missed| False|
+================================+======+======+======+
+--------------------------------+------+------+------+
|    0098_0272_0315_0127_0127.jpg|     0|     1|     5|
+--------------------------------+------+------+------+
|    0099_0096_0119_0148_0148.jpg|     1|     0|     4|
+--------------------------------+------+------+------+
|    0100_0229_0154_0222_0222.jpg|     1|     0|     3|
+--------------------------------+------+------+------+
|                          Total|     2|     1|    12|
+================================+======+======+======+
```

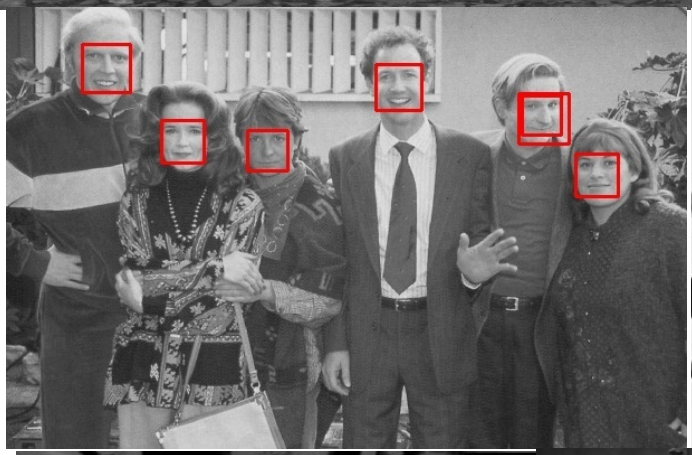
B) Ukázky detekce tváří na testovací sadě CMU

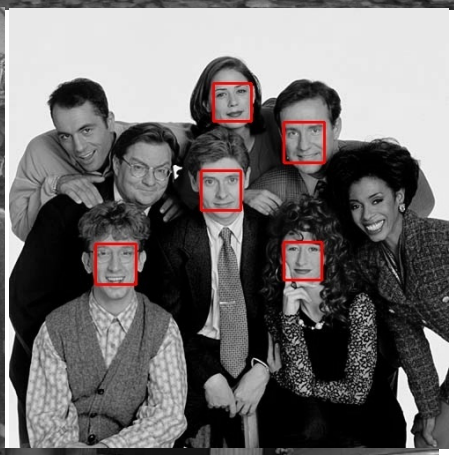
Klasifikátor 1, konfigurace 5

Klasifikátor 2, konfigurace 1



Klasifikátor 3, konfigurace 6





C) Obsah přiloženého DVD

Adresářová struktura

|-- dokumentace

|-- FDetect_spustitelna_aplikace

|-- FDetect_Visual_Studio_2008

|-- trenovaci_sada

|-- trenovani

|-- vysledky

Dokumentace

Obsahuje programátorskou a uživatelskou dokumentaci.

Programátorská dokumentace se spouští souborem *index.html.*

Uživatelská dokumentace se spouští souborem *Uzivatelska_dokumentace.pdf*

FDetect_spustitelna_aplikace

Obsahuje spustitelnou aplikaci a potřebné knihovny pro spuštění. Aplikace se spustí souborem *FDetect.exe*

FDetect_Visual_Studio_2008

Obsahuje vytvořený projekt v prostředí Microsoft Visual Studio 2008 se zdrojovými kódy.

trenovaci_sada

Obsahuje trénovací data.

trenovani

Obsahuje výstupy z procesu trénování.

vysledky

Obsahuje výsledky detekce na testovacích snímcích.